PrimeTime Suite Tool Commands

Version V-2023.12-SP3, April 2024

SYNOPSYS®

Feedback

Contents

1

PrimeTime Suite Tool Commands

This document describes the tool commands supported by the PrimeTime Suite tool.

a

add_command_hook

Add hook into command execution

Syntax

string *add_command_hook* -before *script*

-after *script* -replace *script* [-name *name*] *commandName*

```
string script
string script
string script
string name
string commandName
```

Arguments

`-before script`

Hook runs before command.

`-after script`

Hook runs after command.

`-replace script`

Hook runs in place of command.

`-name name`

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

`commandName`

Command to update.

Feedback

Description

The *add_command_hook* adds a customization point into the execution of an application command. For example, these hooks can be used to either run a report or gather statistics before and after an application command is run.

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

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

Examples

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

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

Force all report_timing calls to use the -nosplit option.

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

See Also

- get_command_hooks
- remove_command_hook
- get_current_hook_command

---

add_eco_scenario_data

Specifies additional ECO data (beyond the *create_scenario* definition) for hybrid view determination by *report_eco_scenarios* and *start_eco_scenarios*.

Syntax

```
status add_eco_scenario_data
-scenario scenario_name
-eco_data directory_name
[-path prefix_list]
```

Feedback

Data Types

```
directory_name  string
prefix_list     list
scenario_name   string
```

Arguments

`-scenario`

Specifies a scenario name for the additional ECO data to be applied to.

`-eco_data`

Specifies the ECO data directory used to determine hybrid views.

`-path` *`prefix_list`*

Specifies a list of relative paths to the instance names of the current design from a higher-level design that includes the current design.

By default, absolute path names are used. Use this option if the ECO data is generated from a higher-level design that includes the current design, and you want to use the higher-level design timing to determine the hybrid views for the current design. If more than one prefix is supplied, all of the instances of the design are considered.

Description

The *add_eco_scenario_data* command is used to import ECO timing data from a source other than the current design analysis.

For example, you might have a top-level design that includes the current design as one of its subblocks, and the top-level analysis has more accurate timing for the interface paths at the block boundary. In this case, use this command to add the ECO data generated at the top level to the current ECO data specified by the *create_scenario* command.

Examples

Suppose the current design 'Block_A' is instantiated as 'Block_A1' and 'Block_A2' at the top level design 'Top'. In other words, the design 'Top' has two sub blocks 'Block_A1' and 'Block_A2'. All designs have two scenarios, 'ss' and 'ff'. The following commands write ECO data from the top-level analysis.

In 'ss' scenario at the top level:

```
pt_shell> write_eco_scenario_data -output top_eco_data/ss
```

In 'ff' scenario at the top level:

```
pt_shell> write_eco_scenario_data -output top_eco_data/ff
```

At the block level, the following commands add the top-level ECO data, assuming the 'Block_A' directory contains ECO data for the current design.

```
pt_shell> create_scenario -name ss -image ss_session -eco_data Block_A/ss
pt_shell> create_scenario -name ff -image ss_session -eco_data Block_A/ff
pt_shell> add_eco_scenario_data -scenario ss \\
            -eco_data top_eco_data/ss \\
            -path { Top/Block_A1 Top/Block_A2 }
pt_shell> add_eco_scenario_data -scenario ff \\
            -eco_data top_eco_data/ff \\
            -path { Top/Block_A1 Top/Block_A2 }
```

See Also

- create_scenario
- report_eco_scenarios
- start_eco_scenarios

---

add_power_state

Adds state information to a supply set or a group or a domain.

Syntax

status *add_power_state*

```
[-supply | -group | -domain] object_name
[-simstate  {simstate}]
[-update]
[-state state_name {[-supply_expr {supply_expression}]
                           [-logic_expr  {logic_expression}]
                           [-simstate    {simstate}] }]
```

Data Types

```
object_name          string
simstate             string
state_name           string
supply_expression    string
logic_expression     string
```

Arguments

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

Specifies the name of a supply set, or the name of a group created with the *create_power_state_group* command or the name of a domain. The name should be a simple (nonhierarchical) name.

`-state state_name`

Specifies the name of the state to add to the supply set or the group or the domain. The name should be a simple (nonhierarchical) name. This option can be specified multiple times in the same command.

`-supply_expr {supply_expression}`

Specifies a Boolean expression in terms of nets and their netstates. This option can only be specified when the *object_name* is a supply set. The only operator allowed in the Boolean expression is && (AND operator). Each subexpression in the Boolean expression specifies the net and netstate definitions using the following syntax:

```
net == netstate
```

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

```
status
{status}
{status nom_voltage}
```

- *status* can be OFF or FULL_ON
- If the *status* is FULL_ON, the voltage value must be defined while assigning status to a net or later with the *-update* option.

`-logic_expr {logic_expression}`

If the *object_name* is a group or a domain, then this option specifies a Boolean expression in terms of supply sets and their states. The only operator allowed in the Boolean expression is &&.

If the *object_name* is a supply set, then this option specifies a SystemVerilog Boolean expression in terms of logic nets and supply nets. The -logic_expr option does not affect implementation and is intended to be used by simulation tools. It is read and ignored by the tool.

`-simstate {simstate}`

Specifies a simstate for the power states associated with a supply set. The -simstate option does not affect implementation and is intended to be used by simulation tools. It is read and ignored by the tool.

`-update`

Updates *object_name* with the state information specified in this command run without overwriting the existing power state definition.

Feedback

Description

This command adds state information to a supply set or a group or a domain.

If the specified object is a supply set, the command sets power states on the nets of a supply set and uses them as supplies for the power state table. If the supply set does not already exist, this command issues an error.

If the specified object is a group, the command creates the power state table by using supply sets that are specified by the -logic_expr option. The group should have been previously created using the *create_power_state_group* command; otherwise, the command issues an error.

When adding power states to a supply set, the *-update* option is supported for adding the new state definition but it's optional. For instance, to add SS1_state1 to the SS1 supply net set, use the following command:

```
prompt > add_power_state SS1 -state SS1_state1 ...
```

To add another power state SS1_state2, use either of the following commands:

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

For adding group power states, the *-update* option is mandatory for adding new state definition. For example, to add A_state2 to GroupA which was set with the A_state1 state, use either of the following commands:

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

In the previous example, the second *add_power_state* command must be run with the *-update* option. Otherwise, the tool issues an error.

The *-update* option is supported for sub-state conjunctions when adding power states to a supply set. For example, you can add power states without the *-update* option:

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

Or use the *-update* option:

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

The tool issues an error when the *-update* option is not specified in the previous example because of duplicate definition of the state HPg.

You can assign a voltage value to the functional net's state with the *-update* option later:

e.g.

```
prompt > add_power_state SS1 \\
         -state HPg {-supply_expr {power =={FULL_ON}}}
prompt > add_power_state SS1 \\
         -state HPg {-supply_expr {power =={FULL_ON 0.8}}} -update
```

The *-update* option is also supported for sub-state conjunctions when adding group power states. For example, you can add group power states without the *-update* option:

```
prompt> add_power_state group_A -group \\
        -state lo2 {-logic_expr {SS1 == HPg && SS2 == HVp}
```

Or use the *-update* option:

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

You can create an empty power state first, and then update it later:

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

Examples

The following example defines a state ON for the supply set SS1 where the supply_expression is a Boolean expression of power and ground.

```
prompt> add_power_state -supply SS1\\
        -state ON  {-supply_expr {power == {FULL_ON 0.8} && ground ==
 {FULL_ON 0}}}
```

The following example defines two states, ON and OFF, for the supply set SS2.

```
prompt> add_power_state -supply SS2\\
        -state ON  {-supply_expr {power == {FULL_ON 0.8} && ground ==
 {FULL_ON 0}}}\\
        -state OFF {-supply_expr {power == {OFF} && ground == {FULL_ON
 0}}}
```

The following shows how to add power states to a PST group PST_GROUP. The system is in state RUN when both supply sets SS1 and SS2 are in state ON; and the system is in state SLEEP when the supply set SS1 is in state ON while the supply set SS2 is in state OFF.

Feedback

```
prompt> create_power_state_group PST_GROUP
prompt> add_power_state -group PST_GROUP \\
        -state RUN   {-logic_expr {SS1 == ON && SS2 == ON}}\\
        -state SLEEP {-logic_expr {SS1 == ON && SS2 == OFF}}
```

See Also

- create_power_state_group
- create_supply_set

---

add_pst_state

Defines the states of each of the supply nets for one possible state of the design.

Syntax

status *add_pst_state*

```
state_name
-pst table_name
-state supply_states
```

Data Types

state_name	string
table_name	string
supply_states	list

Arguments

state_name

Specifies the name of the power state.

-pst *table_name*

Specifies the power state table (PST) to which this state applies.

-state *supply_states*

Lists the supply net state names in the corresponding order of the *-supplies* option listing in the *create_pst* command.

Description

The *add_pst_state* command defines the states of each of the supply nets for one possible state of the design. It is an error if the number of supply state names is different from the number of supply nets within the power state table.

Feedback

Examples

The following example defines the supply net state names for the s1, s2 and s3 power states:

```
create_pst pt -supplies { PN1 PN2 SOC/OTC/PN3 }
add_pst_state s1 -pst pt -state { s08 s08 s08 }
add_pst_state s2 -pst pt -state { s08 s08 off }
add_pst_state s3 -pst pt -state { s08 s09 off }
```

See Also

- create_pst

---

add_to_collection

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

Syntax

collection *add_to_collection*

```
[-unique]
collection1
object_spec
```

Data Types

collection1	collection
object_spec	list

Arguments

`-unique`

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

collection1

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

object_spec

Specifies a list of named objects or collections to add.

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

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

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

Description

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

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

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

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

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

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

Examples

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

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

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

Feedback

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

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

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

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

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

See Also

- append_to_collection
- collections
- query_objects
- remove_from_collection
- sizeof_collection

---

alias

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

Feedback

Syntax

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

Data Types

name	string
def	string

Arguments

name

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

def

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

Description

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

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

Aliases can refer to other aliases.

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

Examples

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

```
prompt> alias q quit
```

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

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

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

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

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

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

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

See Also

- unalias

---

all_clocks

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

Syntax

collection *all_clocks*

Arguments

None.

Description

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

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

Examples

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

pt_shell> *set_propagated_clock [all_clocks]*

See Also

- collections
- create_clock
- derive_clocks

- get_clocks
- set_propagated_clock

---

all_connected

Creates a collection of objects connected to a net, pin, or port object. You can assign this collection to a variable or pass it into another command.

Syntax

collection *all_connected*

```
[-leaf]
object_spec
```

Data Types

object_spec	list

Arguments

`-leaf`

Specifies that the connections of the net that are being returned should be global or leaf pins. When specified, this gives the leaf pins of a hierarchical net. For non-hierarchical nets, there is no difference in output.

object_spec

Specifies the object whose connections are returned. This is a collection of one element which is a net, pin, or port collection, or the name of a net, pin, or port.

Description

The *all_connected* command creates a collection of objects connected to a specified net, pin, or port. The *object_spec* option is either a collection of exactly one net, pin, or port object, or a name of an object. If it is a name, PrimeTime searches for a net, pin, or port, in that order. If the *object_spec* refers to a net, you can use the *-leaf* option to get the global leaf pins of the net.

The command returns a collection. The collection can contain nets, ports, pins, or a combination of ports and pins. In the latter case, the ports are first and they are followed by the pins.

When issued from the command prompt, the *all_connected* command behaves as though the *query_objects* command has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

Feedback

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

Examples

The following example shows all objects connected to net "CLOCK":

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

See Also

- collections
- get_nets
- get_pins
- query_objects
- collection_result_display_limit

---

all_fanin

Creates a collection of pins, ports, or cells in the fanin of specified objects.

Syntax

collection *all_fanin*

```
[-from from_list]
[-through through_list]
-to to_list
[-startpoints_only]
[-only_cells]
[-flat]
[-quiet]
[-step_into_hierarchy]
[-levels level_count]
[-pin_levels pin_count]
[-trace_arcs timing | enabled | all]
[-continue_trace pin_types]
```

Data Types

```
from_list        list
through_list     list
to_list          list
level_count      int
pin_count        int
pin_types        string
```

Feedback

Arguments

`-from` *`from_list`*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanin of each object in *to_list* becomes part of the resulting collection only if it is in the fanout cone of at least one object in *from_list*. If a net is specified, the effect is the same as listing all load pins on the net.

`-through` *`through_list`*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. If a *through_list* is specified, the fanin of each object in *to_list* is restricted to objects on paths through the pins, ports or nets in *through_list*.

`-to` *`to_list`*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanin of each object in *to_list* becomes part of the resulting collection. If a net is specified, the effect is the same as listing all driver pins on the net. This argument is required.

`-startpoints_only`

Includes only the timing startpoints in the result.

`-only_cells`

Includes only cells in the timing fanin of the *to_list* and not pins or ports.

`-flat`

There are two major modes in which this command functions: hierarchical (the default) and flat. When in hierarchical mode, only objects within the same hierarchical level as the current *to_list* object are included in the result. In flat mode, the relevant leaf objects through design hierarchy are included; the only non-leaf objects in the result are any specified hierarchical *to_list* objects.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-step_into_hierarchy`

You can use this option only in hierarchical mode and with either the *-levels* or *-pin_levels* option. Without this option, a hierarchical block at the same level of hierarchy as the current object is considered to be a cell; the input pins are considered a single level away from the related output pins, regardless of what is inside the block.

With the switch enabled, the counting is performed as though the design were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related output pins.

`-levels` *`level_count`*

The traversal stops when reaching a depth of search of *cell_count* hops, where the counting is performed over the layers of cells of same distance from the object in *to_list*.

`-pin_levels` *`pin_count`*

The traversal stops when reaching a depth of search of *pin_count* hops, where the counting is performed over the layers of pins of same distance from the object in *to_list*.

`-trace_arcs timing | enabled | all`

Specifies the type of combinational arcs to trace during the traversal:

- *timing* (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- *enabled* - Permits the tracing of all enabled arcs and disregards case analysis values.
- *all* - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

`-continue_trace` *`pin_types`*

This option permits the user to override the default behavior of the traversal which is to stop at all timing startpoints. When used, the travsal is permitted through startpoint pins of a type specified with *pin_types*. Allowed *pin_types* are:

- *generated_clock_source* - Specifies that the traversal is to continue tracing through source pins of generated clocks including sequential clock pins in the generated clock source network.

Description

The *all_fanin* command creates a collection of objects in the timing fanin of specified "to" pins, ports, or nets in the design. A pin is considered to be in the timing fanin of an object in *to_list* if there is a timing path through combinational logic from the pin to that object (see also the *-trace_arcs* option). The fanin stops at input ports and the clock pins of registers (sequential cells).

If the current instance in the design is not the top level of hierarchy, only objects within the current instance are returned.

The fanin cone can be topologically restricted by using the *-from* and *-through* options.

Feedback

Examples

This example shows the timing fanin of a port in the design. The fanin includes a register, reg1.

```
pt_shell> query_objects [all_fanin -to out_1]
{"out_1", "reg1/Q", "reg1/CP"}
```

This example shows the flat mode of *all_fanin*. The object in *to_list* is an input pin of a hierarchical cell, H1, which is connected to an output pin of another hierarchical cell, H2. H2 contains additional hierarchy and eventually, a leaf cell with 2 inputs, each of which has a top level register in its fanin.

```
pt_shell> query_objects [all_fanin -to H1/a -flat]
{"H1/a", "H2/U1/n1/Z", "H2/U1/n1/A", "H2/U1/n1/B",
 "reg1/Q", "reg2/Q", "reg1/CP", "reg2/CP"}
```

This example shows the timing fanin of an output port that passes through pin U1/Z or pin U2/Z.

```
pt_shell> query_objects [all_fanin -to out -through {U1/Z U2/Z}]
{"out", "U3/Z", "U3/A", "U3/B", "U1/Z", "U1/A", "U2/Z", "U2/Z",
 "reg1/Q", "reg2/Q", "reg1/CP", "reg2/CP"}
```

This example shows the timing fanin of an output port the passes through pin U1/Z and pin U3/A.

```
pt_shell> query_objects [all_fanin -to out -through U1/Z -through U3/A]
{"out", "U3/Z", "U3/A", "U1/Z", "U1/A", "reg1/Q", "reg1/CP"}
```

See Also

- all_fanout
- current_instance
- report_transitive_fanin
- report_transitive_fanout

---

all_fanout

Creates a collection of pins, ports, or cells in the fanout of the specified objects.

Syntax

collection *all_fanout*

```
-from from_list
   | -clock_tree
[-through through_list]
```

```
[-to to_list
[-endpoints_only]
[-only_cells]
[-flat]
[-quiet]
[-step_into_hierarchy]
[-levels level_count]
[-pin_levels pin_count]
[-trace_arcs timing | enabled | all]
[-continue_trace pin_types]
```

Data Types

from_list	list
through_list	list
to_list	list
level_count	int
pin_count	int
pin_types	string

Arguments

`-from` *from_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanout of each object in *from_list* becomes part of the resulting collection. If a net is specified, the effect is the same as listing all load pins on the net.

This option is mutually exclusive with the *-clock_tree* option. At least one must be specified.

`-through` *through_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. If a *through_list* is specified, the fanout of each object in *from_list* is restricted to objects on paths through the pins, ports or nets in *through_list*.

`-to` *to_list*

Specifies a list of pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanout of each object in *from_list* becomes part of the resulting collection only if it is in the fanin cone of at least one object in *to_list*. If a net is specified, the effect is the same as listing all driver pins on the net.

`-clock_tree`

Specifies that all clock source pins and ports in the design are used as objects in the *from_list*. Clock sources are specified using *create_clock*. If there are

no clocks, or if the clocks have no sources, the command returns an empty collection. This option constrains the search to objects in the clock network only.

This option is mutually exclusive with the *-from* option. At least one must be specified.

`-endpoints_only`

Includes only the timing endpoints in the result.

`-only_cells`

Includes only cells in the timing fanout of the *from_list* and not pins or ports.

`-flat`

There are two major modes in which this command functions: hierarchical (the default) and flat. When in hierarchical mode, only objects within the same hierarchical level as the current *from_list* object are included in the result. In flat mode, the relevant leaf objects through design hierarchy are included; the only non-leaf objects in the result are any specified hierarchical *from_list* objects.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-step_into_hierarchy`

You can use this option only in hierarchical mode and with either the *-levels* or *-pin_levels* option. Without this option, a hierarchical block at the same level of hierarchy as the current object is considered to be a cell; the output pins are considered a single level away from the related input pins, regardless of what is inside the block. With the switch enabled, the counting is performed as though the design were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related input pins.

`-levels` *`level_count`*

The traversal stops when reaching a depth of search of *cell_count* hops, where the counting is performed over the layers of cells of same distance from the object in *from_list*.

`-pin_levels` *`pin_count`*

The traversal stops when reaching a depth of search of *pin_count* hops, where the counting is performed over the layers of pins of same distance from the object in *from_list*.

`-trace_arcs timing | enabled | all`

Specifies the type of combinational arcs to trace during the traversal:

- *timing* (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- *enabled* - Permits the tracing of all enabled arcs and disregards case analysis values.
- *all* - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

`-continue_trace` *`pin_types`*

This option permits the user to override the default behavior of the traversal which is to stop at all timing endpoints. When used, the travsal is permitted through endpoint pins of a type specified with *pin_types*. Allowed *pin_types* are:

- *generated_clock_source* - Specifies that the traversal is to continue tracing through source pins of generated clocks including sequential clock pins in the generated clock source network.

Description

The *all_fanout* command creates a collection of objects in the timing fanout of specified "from" pins, ports, or nets in the design. A pin is considered to be in the timing fanout of an object in *from_list* if there is a timing path through combinational logic from that object to the pin (see also the *-trace_arcs* option). The fanout stops at output ports and the inputs to registers (sequential cells). The from objects are specified using either *-clock_tree* or *-from from_list*.

If the current instance in the design is not the top level of hierarchy, only objects within the current instance are returned.

The fanout cone can be topologically restricted by using the *-through* and *-to* options.

Examples

This example shows the timing fanout of a port in the design. The fanout includes the reg3 register.

```
pt_shell> query_objects [all_fanout -from in1]
{"in1", "reg3/D"}
```

This example shows the difference between the hierarchical and flat modes of *all_fanout*. The object in the *from_list* is the H3/z1 output pin of a hierarchical cell, which is connected to the H4/a input pin of another hierarchical cell. H4 contains the U1 leaf cell with input A and output Z. The first command is in hierarchical mode and shows that hierarchical pins are included in the result. The second command is in leaf mode, and leaf pins from the lower level are included.

```
pt_shell> query_objects [all_fanout -from H3/z1]
{"H3/z1", "H4/a", "H4/z", "reg2/D"}

pt_shell> query_objects [all_fanout -from H3/z1 -flat]
{"H3/z1", "H4/U1/A", "H4/U1/Z", "reg2/D"}
```

This example shows how to get the fanout of reg1/CP on paths passing through u1/Z and U3/Z.

```
pt_shell> query_objects [all_fanout -from reg1/CP -through U1/Z \\
          -through U3/A]
{"out", "U3/Z", "U3/A", "U1/Z", "U1/A", "reg1/Q", "reg1/CP"}
```

See Also

- all_fanin
- current_instance
- report_transitive_fanin
- report_transitive_fanout

---

all_inputs

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

Syntax

collection *all_inputs*

```
[-level_sensitive]
[-exclude_clock_ports]
[-edge_triggered]
[-clock clock_name]
```

Data Types

clock_name list

Arguments

`-level_sensitive`

Only considers ports with level-sensitive input delay. This is specified by the *set_input_delay 2 -clock CLK -level_sensitive IN1* command.

`-exclude_clock_ports`

Excludes input ports which serve as clock sources.

`-edge_triggered`

Only considers ports with edge-triggered input delay. This is specified by *set_input_delay 2 -clock CLK IN2*.

`-clock` *`clock_name`*

Only considers ports with input delay relative to a specific clock. This can be the name of a clock, or a collection containing a clock.

Description

The *all_inputs* command creates a collection of all input or inout ports in the current design. You can limit the contents of the collection by specifying the type of input delay that must be on a port.

You can remove clock source ports from the resulting collection by specifying *-exclude_clock_ports* option.

If you want only certain ports, use the *get_ports* command to create a collection of ports matching a specific pattern and optionally passing filter criteria.

When issued from the command prompt, the *all_inputs* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

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

Examples

The following example specifies a driving cell for all input ports.

```
pt_shell> set_driving_cell -lib_cell FFD3 -pin Q [all_inputs]
```

See Also

- collections
- get_ports
- report_port
- query_objects
- set_driving_cell
- set_input_delay
- collection_result_display_limit

Feedback

all_instances

Creates a collection of all instances of a specific design or library cell in the current design, relative to the current instance. You can assign the resulting collection of cells to a variable or pass it into another command.

Syntax

collection *all_instances*

```
[-hierarchy]
object_spec
```

Data Types

```
object_spec     string
```

Arguments

`-hierarchy`

Searches for instances in all levels of instance hierarchy below the current instance. By default, only instances from the current level of hierarchy are considered.

object_spec

Specifies the target design or library cell. This can be a design collection, lib_cell collection, or name.

Description

The *all_instances* command creates a collection of cells that are instances of a design or library cell. The search for instances is made relative to the current instance within the current design. By default, *all_instances* considers only instances at the current level of the hierarchy. If you use the *-hierarchy* option, the search continues throughout the hierarchy.

The *object_spec* can be a simple name. In this case, any instance of a design or lib_cell with that name will match. Alternatively, the *object_spec* can be a collection of exactly one design or one library cell. Any other collection results in an error message. Using the collection can help focus the search for instances of specific designs or lib_cells, especially after *swap_cell* has been used.

When issued from the command prompt, *all_instances* behaves as though *query_objects* has been called to display the objects in the collection. By default, a maximum of 100 objects is displayed. You can change this maximum using the *collection_result_display_limit* variable.

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

Feedback

Examples

The following example uses *all_instances* to get the instances of the design 'low' in the current level of hierarchy.

```
pt_shell> all_instances low
{"U1", "U3"}
```

The following example uses *-hierarchy* to display instances of a design across multiple levels of hierarchy.

```
pt_shell> query_objects [all_instances low -hierarchy]
{"U1", "U2/U1", "U3"}
```

In the following example, there are two instances of design "inter". One of them is swapped for a new design. The difference between using *all_instances* with a name and a collection of one design is shown.

```
pt_shell> swap_cell i1 inter_2.db:inter
 ...unlinking i1
1
pt_shell> all_instances inter
{"i1", "i2"}
pt_shell> all_instances [get_designs inter_2.db:inter]
{"i1"}
```

See Also

- collections
- current_design
- current_instance
- get_cells
- get_designs
- get_lib_cells
- list_designs
- query_objects
- collection_result_display_limit

---

all_outputs

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

Feedback

Syntax

collection *all_outputs*

```
[-level_sensitive]
[-edge_triggered]
[-clock clock_name]
```

Data Types

clock_name	list

Arguments

`-level_sensitive`

Only considers ports with level-sensitive output delay. This is specified by *set_output_delay 2 -clock CLK -level_sensitive IN1*.

`-edge_triggered`

Only considers ports with edge-triggered output delay. This is specified by *set_output_delay 2 -clock CLK IN2*.

`-clock clock_name`

Only considers ports with output delay relative to a specific clock. This can be the name of a clock, or a collection containing a clock.

Description

The *all_outputs* command creates a collection of all output or inout ports in the current design. You can limit the contents of the collection by specifying the type of output delay that must be on a port.

If you want only certain ports, use *get_ports* to create a collection of ports matching a specific pattern and optionally passing filter criteria.

When issued from the command prompt, *all_outputs* behaves as though *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the variable *collection_result_display_limit*.

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

Examples

The following example specifies a pin capacitance for all output ports.

```
pt_shell> set_load 4.56 [all_outputs]
```

Feedback

The following example shows the names all of the output ports with output delay relative to PHI1.

```
pt_shell> all_outputs -clock PHI1
```

See Also

- collections
- get_ports
- report_port
- query_objects
- set_output_delay
- collection_result_display_limit

---

all_registers

Creates a collection of register cells or pins. You can assign the resulting collection to a variable or pass it to another command.

Syntax

collection *all_registers*

```
[-clock clock_name]
[-rise_clock rise_clock_name]
[-fall_clock fall_clock_name]
[-cells]
[-data_pins]
[-clock_pins]
[-slave_clock_pins]
[-async_pins]
[-output_pins]
[-level_sensitive]
[-edge_triggered]
[-master_slave]
[-no_hierarchy]
```

Data Types

```
clock_name              list
rise_clock_name         list
fall_clock_name         list
```

Arguments

`-clock` *`clock_name`*

Considers all registers clocked by *clock_name*. This is either the name of a clock, or a collection containing a clock or clocks. For example, all registers whose clock pins are in the fanout of the specified clock or clocks.

`-rise_clock` *`rise_clock_name`*

Considers all registers clocked by *rise_clock_name* and having open edge effectively the rising clock edge. This is either the name of a clock, or a collection containing a clock. For example, rising edge triggered flip-flops without any inversion on the clock path or falling edge triggered flip-flops with inversion on the clock path.

`-fall_clock` *`fall_clock_name`*

Considers all registers clocked by *fall_clock_name* and having open edge effectively falling the clock edge. This is either the name of a clock, or a collection containing a clock. For example, rising edge triggered flip-flops with inversion on the clock path or falling edge triggered flip-flops without any inversion on the clock path.

`-cells`

Collects cells; this is the default behavior of this command. The cells are registers and are further limited by other command options.

`-data_pins`

Collects register data pins. The collection can be limited by other command options.

`-clock_pins`

Collects register clock pins. The collection can be limited by other command options.

`-slave_clock_pins`

Collects register slave clock pins (the slave clock pins of master-slave registers). Specify slave clock pins as *clocked_on_also* in the library.

`-async_pins`

Collects asynchronous preset or clear pins.

`-output_pins`

Collects register output pins.

`-level_sensitive`

Considers only level-sensitive latches.

`-edge_triggered`

Considers only edge-triggered flip-flops.

`-master_slave`

Considers only master or slave register cells.

`-no_hierarchy`

Considers only the current instance; does not descend the hierarchy.

Description

The *all_registers* command creates a collection of pins or cells related to registers.

When issued from the command prompt, *all_registers* behaves as though *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; to change this maximum, set the *collection_result_display_limit* variable.

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

Examples

This example performs a trace from a port to all register clock pins which are clocked by CLK.

```
pt_shell> report_timing -from [get_ports {CLK}] \\
          -to [all_registers -clock CLK -clock_pins]

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : counter
 ...
****************************************

  Startpoint: CLK (input port clocked by CLK)
  Endpoint: ffa/CP (internal pin)
  Path Group: (none)
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  input external delay                     0.00       0.00 r
  CLK (in)                                 0.00       0.00 r
  ffa/CP (FD2)                             0.00       0.00 r
  data arrival time                                   0.00
  -----------------------------------------------------------
  (Path is unconstrained)
```

Feedback

The following example performs a trace from all registers clock pins, which are triggered by clock rising edge.

```
pt_shell> report_timing -from [all_registers -rise_clock \\
            CLK -clock_pins]

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : li_FD3x
 ...
****************************************

  Startpoint: ff3 (falling edge-triggered flip-flop clocked by clk')
  Endpoint: ff4 (falling edge-triggered flip-flop clocked by clk')
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk' (fall edge)                   0.00       0.00
  clock network delay (ideal)              0.00       0.00
  ff3/CP (FD3x)                            0.00       0.00 f
  ff3/Q (FD3x)                             1.44       1.44 f
  iv2/Z (IVA)                              0.31       1.75 r
  ff4/D (FD3x)                             0.00       1.75 r
  data arrival time                                   1.75

  clock clk' (fall edge)                  10.00      10.00
  clock network delay (ideal)              0.00      10.00
  ff4/CP (FD3x)                                      10.00 f
  library setup time                      -0.90       9.10
  data required time                                  9.10
  -----------------------------------------------------------
  data required time                                  9.10
  data arrival time                                  -1.75
  -----------------------------------------------------------
  slack (MET)                                         7.35
```

See Also

- collections
- get_cells
- get_pins
- collection_result_display_limit

Feedback

annotate_trace

Control annotation of traced command execution to the shell output.

Syntax

annotate_trace [-start|-stop] [-profile profile_annotation_type] [-log log_string] [-quiet]

string *log_string* string *profile_annotation_type*

Arguments

`-start`

This option is mutually exclusive with -stop. The option starts the annotation of traced command execution to the shell output. Each executed command string is annotated to the output prepended with ";## ".

`-stop`

This option is mutually exclusive with -start. This option stops annotation of traced command execution to the shell output.

`-profile` *`profile_annotation_type`*

This option selects the profile data output behavior for command annotation. This option is only meaningful when given with the -start option. The argument is one of: on, with_start, off. The /fIon/fP value starts annotation with profile data output enabled in the default mode which is to output the data only at the end of a command invocation. The data is output only if the delta values from the beginning of the command invocation meet or exceed the thresholds set for each of the reported metrics. The /fIwith_start/fP value starts annotation with profile data output both at the beginning and at the end of a command invocation. The profile data is output at the beginning irregardless of any thresholds since delta values cannot be calculated at the beginning of the invocation. The profile data output at the end of command invocation is controlled by thresholds. The /fIoff/fP value disables profile data output. If profile data annotation is enabled but profile metric gathering has not been enabled with set_trace_option, all profile metrics are auto-enabled. If omitted, annotation is started without profile data output.

`-log` *`log_string`*

This option outputs the given log_string to the shell output prepended with ";## ". If the given string contains newline characters, each line is prendended with ";## ".

Feedback

`-quiet`

If given, this option causes the command to ignore error conditions such as an attempt to log a comment string before annotation is started. The command will exit quietly when it encounters an error condition.

Description

The *annotate_trace* command performs operations related to annotating traced command execution to the shell output. If the application creates an output log, the annotation is made also to the output log. The command strings which are annotated to the output are those that are traced for creating a traced command replay log.

The *annotate_trace* with no options will report on the current status of annotation: on or off.

Examples

The following example starts annotation of executed commands to the output. Then the command is invoked again with no options to report on the status.

```
prompt> annoate_trace -start
prompt> annoate_trace
on
```

The following example configures profile metrids, then starts annotation of executed commands to the output along with the profile data.

```
prompt> set_trace_topion -profile all
prompt> set_trace_topion -memory_threshold 10
prompt> set_trace_topion -cpu_threshold 0
prompt> annotate_trace -start -profile with_start
```

See Also

- set_trace_option
- get_trace_option
- log_trace

---

append_to_collection

Adds objects to a collection and modifies a variable.

Syntax

collection *append_to_collection* [-dict *dict_var*]

```
[-unique]
var_name
object_spec
```

Feedback

Data Types

var_name	variable referencing a collection
dict_var	variable referencing a dict
object_spec	list

Arguments

`-unique`

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

var_name

Specifies a variable name. The objects matching *object_spec* are added into the collection referenced by this variable.

`-dict` *dict_var*

Specifies a variable name referencing a dict object. When this is specified the *var_name* argument specifies the key in the dict to update.

object_spec

Specifies a list of named objects or collections to add.

Description

The *append_to_collection* command allows you to add elements to a collection. This command treats the variable name given by the *var_name* option as a collection, and it appends all of the elements in *object_spec* to that collection. If the variable does not exist, it is created as a collection with elements from the *object_spec* as its value. If the variable does exist and it does not contain a collection, it is an error.

Alternatively this command may also update a value in a dict. When *-dict* is specified the *dict_var* is treated as a dict and the key specified by *var_name* is updated. If the dict variable is unset the dict is created. If the value referenced by the variable is not a dict it is an error.

The result of the command is the collection that was initially referenced by the *var_name* option (or dict key value), or the collection created if the variable did not exist.

The *append_to_collection* command provides the same semantics as the common use of the *add_to_collection* command; however, this command shows significant improvements in performance.

An example of replacing the *add_to_collection* command with the *append_to_collection* command is provided below. For example,

```
set var_name [add_to_collection $var_name $objs]
```

Feedback

Using the *append_to_collection* command, the command becomes:

```
append_to_collection var_name $objs
```

Updating a key in a dictionary:

```
append_to_collection -dict drivers $pinName $newDrivers
```

The *append_to_collection* command can be much more efficient than the *add_to_collection* command if you are building up a collection in a loop. The arguments of the command have the same restrictions as the *add_to_collection* command. For more information about these restrictions, see the *add_to_collection* man page.

Examples

The following example from PrimeTime shows how a collection can be built up using the *append_to_collection* command:

```
pt_shell> set xports
Error: can't read "xports": no such variable
        Use error_info for more info. (CMD-013)
pt_shell> append_to_collection xports [get_ports in*]
{"in0", "in1", "in2"}
pt_shell> append_to_collection xports CLOCK
{"in0", "in1", "in2", "CLOCK"}
```

See Also

- add_to_collection
- foreach_in_collection
- index_collection
- remove_from_collection
- sizeof_collection

---

apply_power_model

Binds power models to instances in the design and connects the interface supply set handles and logic ports of a previously loaded power model.

Syntax

string *apply_power_model*

Feedback

```
[model_name]
[-elements instance_name_list]
[-supply_map {{instance_scope_supply_set current_scope_supply_set}*}]
[-port_map {{instance_scope_logic_port current_scope_logic_net}*}]
[-parameters {{power_model_ parameter design_object}*}]
```

Data Types

```
instance_name_list                                  list
{instance_scope_supply_set current_scope_supply_set}  list
{instance_scope_logic_port current_scope_logic_net}   list
{power_model_ parameter design_object}              list
```

Arguments

`model_name` *string*

PrimeTime reads and ignore this option. Complete support will be available in future releases.

`-elements` *instance_name_list*

PrimeTime reads and ignore this option. Complete support will be available in future releases.

`-supply_map` *{{instance_scope_supply_set current_scope_supply_set}*}*

PrimeTime reads and ignore this option. Complete support will be available in future releases.

`-port_map` *{{instance_scope_logic_port current_scope_logic_net}*}*

PrimeTime reads and ignore this option. Complete support will be available in future releases.

`-parameters` *{{power_model_ parameter design_object}*}*

PrimeTime reads and ignore this option. Complete support will be available in future releases.

Description

PrimeTime reads and ignore this command. Complete support will be available in future releases.

Examples

The following command instantiates power model named MODEL.

```
prompt> apply_power_model MODEL -elements {.}
1
```

Feedback

See Also

- define_power_model

---

apropos

Searches the command database for a pattern.

Syntax

string *apropos*

```
[-symbols_only]
pattern
```

Data Types

```
pattern      string
```

Arguments

`-symbols_only`

Searches only command and option names.

pattern

Searches for the specified *pattern*.

Description

The *apropos* command searches the command and option database for all commands that contain the specified *pattern*. The *pattern* argument can include the wildcard characters asterisk (*) and question mark (?). The search is case-sensitive. For each command that matches the search criteria, the command help is printed as though *help -verbose* was used with the command.

Whereas *help* looks only at command names, *apropos* looks at command names, the command one-line description, option names, and option value-help strings. The search can be restricted to only command and option names with the *-symbols_only* option.

When searching for dash options, do not include the leading dash. Search only for the name.

Examples

In the following example, assume that the *get_cells* and *get_designs* commands have the *-exact* option. Note that without the *-symbols_only* option, the first search picks up commands which have the string "exact" in the one-line description.

```
prompt> apropos exact
 get_cells            # Create a collection of cells
```

```
    [-exact]                (Wildcards are considered as plain characters)
    patterns                (Match cell names against patterns)

  get_designs           # Create a collection of designs
    [-exact]                (Wildcards are considered as plain characters)
    patterns                (Match design names against patterns)

  real_time             # Return the exact time of day

prompt> apropos exact -symbols_only
  get_cells             # Create a collection of cells
    [-exact]                (Wildcards are considered as plain characters)
    patterns                (Match cell names against patterns)

  get_designs           # Create a collection of designs
    [-exact]                (Wildcards are considered as plain characters)
    patterns                (Match design names against patterns)
```

See Also

- help

---

as_collection

Find a collection by name

Syntax

string *as_collection* [-check] *collection1*

```
string collection1
```

Arguments

`-check`

Return 1 if collection1 is a collection

collection1

The collection to convert

Description

This command looks up a string value and returns it as a collection it the collection is still live. All other values are passed through this command unmodified.

This command is also provides a simple way to check if the input is a collection.

Feedback

Examples

Check if a value is a collection:

```
shell> as_collection -check [get_cells]
1
shell> as_collection -check hello
0
```

See Also

- collections
- foreach_in_collection

---

associate_supply_set

Associates a supply set handle to another supply set handle or supply set reference.

Syntax

status *associate_supply_set*

```
-handle supply_set_handle
supply_set_name
```

Data Types

```
supply_set_handle     string
supply_set_name       string
```

Arguments

`-handle` *supply_set_handle*

Specifies the supply set handle on which the action is performed.

This is a required argument.

supply_set_name

Specifies the name of the supply set to which the supply set handle is associated. The name can be the name of a supply set or a supply set handle.

This is a required argument.

Description

The *associate_supply_set* command associates the supply set handle specified with the *-handle* option to the specified supply set. After the association, the tool treats corresponding functions in each supply set to be virtually connected. Therefore, the supply

sets inherit common switching behavior and must eventually be resolved to the same physical supply net.

A supply set is an abstract collection of supply nets, consisting of two supply functions, power and ground.

Examples

The following example associates primary supply handle of domain PD1 to the SS1 supply set.

```
prompt> associate_supply_set SS1 -handle PD1.primary
1
```

See Also

- create_power_domain
- create_supply_set

C

---

cache_distributed_attribute_data

Collects and stores remotely accessible attribute data locally in the manager process.

Syntax

string *cache_distributed_attribute_data*

```
[-class class_name]
[-attributes attribute_list]
```

Data Types

class_name	string
attribute_list	list

Arguments

`-class` *class_name*

Specifies the type of object: *port*, *cell*, *pin*, *net*. Use this option to specify the name of the class/object.

`-attributes` *attribute_list*

Store only the specified list of attributes for each object in the manager process.

Feedback

Description

The *cache_distributed_attribute_data* stores the value for the specified attributes locally in the manager process, such that when any attribute query is made for the remotely accessible attributes, the attributes will be available locally in the manager process. The command is supported in *Distributed Static Timing Analysis (DSTA)* and *Distributed Multi-Scenario Analysis (DMSA)* mode.

Examples

The following example shows the usage of cache_distributed_attribute_data command in Distributed Static Timing Analysis *(DSTA)* mode.

```
pt_shell> cache_distributed_attribute_data -class pin -attributes
 {min_rise_slack max_rise_slack}

Start of Manager/Worker Task Processing
----------------------------------------
Started    : Task execution on 'partition2' (Hid=2)
Started    : Task execution on 'partition4' (Hid=1)
Started    : Task execution on 'partition1' (Hid=3)
Started    : Task execution on 'partition0' (Hid=4)
Started    : Task execution on 'partition3' (Hid=5)
Successful : Task execution on 'partition3' (Hid=5)
Successful : Task execution on 'partition0' (Hid=4)
Successful : Task execution on 'partition1' (Hid=3)
Successful : Task execution on 'partition2' (Hid=2)
Successful : Task execution on 'partition4' (Hid=1)
----------------------------------------
End of Manager/Worker Task Processing
```

```
The following example shows the usage of cache_distributed_attribute_data
 command in
Distributed Multi-Scenario Analysis (DMSA) mode.

pt_shell> cache_distributed_attribute_data -class pin -attributes
 {min_rise_slack max_rise_slack}

Start of Manager/Worker Task Processing
-----------------------------------
Started    : Command execution in scenario 'scen1'
Started    : Command execution in scenario 'scen2'
Succeeded  : Command execution in scenario 'scen2'
Succeeded  : Command execution in scenario 'scen1'
-----------------------------------
End of Manager/Worker Task Processing

Restoring Attribute data at Manager
```

Feedback

cell_of

Creates a collection of cells of the given pins. The *cell_of* command is a DC Emulation command provided for compatibility with Design Compiler.

Syntax

string *cell_of*

```
        object_list
```

Data Types

object_list	list

Arguments

object_list

Creates a list of pins for which cells to get.

Description

The *cell_of* command creates a collection of cells owned by a list of pins. The command exists in PrimeTime for compatibility with Design Compiler. Complete information about the *cell_of* command can be found in the Design Compiler documentation. The supported method for getting pins of cells is to use the *get_cells* command with the *-of_objects* option.

See Also

- get_cells

change_selection

Changes the selection in the GUI, taking a collection of objects and changing the selection according to the type of change specified.

Syntax

int *change_selection*

```
[-name slct_bus]
[-replace]
[-add]
[-remove]
[-toggle]
[-push]
[-undo]
```

```
[-type object_type]
collection
```

Data Types

```
slct_bus              string
object_type           list
collection            list
```

Arguments

`-name slct_bus`

Specifies to change the selection bus by using the value of *slct_bus*. By default, the command changes the selection bus by using the name *global*.

`-replace`

Replaces the current selection with the objects in the collection. This is the default behavior of the command if you do not specify any optional parameter.

`-add`

Adds the objects in the collection to the current selection. By default, this option is off.

`-remove`

Removes the objects that are specified in the collection from the current selection. By default, this option is off.

`-toggle`

Adds each item that is specified in the collection to the selection bus if it is not currently contained there. If it is currently contained in the selection bus, remove it. By default, this option is off.

`-push`

Pushes current selection onto selection stack before changing the selection.

`-undo`

Undo to previous selection.

Undo is single level so running undo twice will return to original selection.

`-type object_type`

Specifies the type to change. Only those items from the collection that are of the type specified by *object_type* are used to change the selection. The valid values are *design*, *port*, *net*, *cell*, *pin*, and *path* (timing path). By default, the command uses the entire collection.

collection

Specifies the collection of objects to use to change the selection. The type of change that is applied to the current selection with the *collection* is specified by the optional parameters listed above. By default, this option is off.

Description

The *change_selection* command changes the selection in the GUI. When selections are changed, the GUI updates all relevant windows to reflect it.

A collection of objects and the type of change are given as input to the command. The collection of objects might be returned as the result of another command such as the *get_designs* command. If the collection is empty and you use the *-replace* option (or let the command default by specifying no option), the current selection is cleared.

If you use the *-type* option, only the type of objects specified are used to change the current selection. For example, if you use *-type design*, the command uses only the design objects in the collection to change the current selection. If you do not use *-type* option, all objects in the collection are used to change the current selection. For example, if you use the *-add* option without using the *-type* option, all objects, regardless of their type, are added to the current selection.

For information about collections, see the *collections* man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example replaces the current selection with the collection of design objects, regardless of type.

```
prompt> change_selection [get_designs]
```

The following example adds a cell named U5 to the selection made in the example above.

```
prompt> change_selection -add [get_cells U5]
```

The following example removes all pin objects from the current selection.

```
prompt> change_selection -remove -type pin [get_selection]
```

The following example clears the selection.

```
prompt> change_selection ""
```

The following example creates a new selection bus and adds a net named n1 to the selection.

Feedback

```
prompt> set slct [create_selection_bus]
prompt> change_selection  -name $slct [get_nets 1]
```

See Also

- collections
- filter
- filter_collection
- get_selection
- gui_selection_stack
- query_objects

---

characterize_context

Captures the timing context of a list of instances for hierarchical analysis.

Syntax

string *characterize_context*

```
-block block_name
[-top]
[-instances cell_list]
[-name mim_group_name]
[-mim_reference mim_ref_instance]
```

Data Types

```
block_name       string
cell_list        list
mim_group_name   string
mim_ref_instance string
```

Arguments

`-block block_name`

Specifies the name of the reference subblock design. One or more instances of this block are characterized.

`-top`

Specifies top design is characterized for top only context writing. In top only mode, the tool only writes out the top level constraints and excludes all constraints from characterized sub-instances.

`-instances` *`cell_list`*

Specifies a list of one or more instances in the current design to be characterized. They must be instances of the reference block specified by the *-block* option.

If you omit this option, the command tries to merge all instances of the block specified by the *-block* option. If all the instances cannot be merged due to differences in context, the command merges the largest mergeable cluster of instances with similar context.

`-name` *`mim_group_name`*

Specifies the name of the instance group subset of multiply instantiated modules (MIMs). The instances specified in *-instances* have their context merged for a single common block-level analysis. To perform merging, using the *-name* option is mandatory if *-instances* is used, unless the given instances cover all the instances of the given block.

`-mim_reference` *`mim_ref_instance`*

Specifies the instance name of the MIM reference used for context merging. By default, MIM context merging automatically chooses a reference instance from the largest mergeable instance cluster. With this option, the specified instance is used for context merging, possibly overriding the default.

Description

The *characterize_context* command captures the timing context of instances of subdesigns from the chip-level timing environment to support hierarchical analysis. You can use the characterization data to analyze the timing behavior of the block instance by itself, in the context of the top-level design where it is used.

The timing context of an instance includes waveforms of the clocks affecting this instance, input arrival times, output required times, timing exceptions, design rules, propagated constants, input drives, and capacitive loads. You can report the context data using the *report_context* command.

To write out the context information for other PrimeTime sessions or other tools, use the *write_context* command.

Examples

The following script derives, reports, and writes out the context for instance U12 of block BLKA. The next PrimeTime session reads in the block and block context, and then analyzes the timing of the block by itself in the context of top-level design where the block is used.

```
characterize_context -block blkA -instances U12
report_context U12
write_context -format psth -output U12_context [get_cells U12]
```

```
------------------------

read_verilog block BLKA.v
link_design BLKA
 ...
read_context U12_context
update_timing
```

The following script characterizes the context for four instances of the block BLKA. It merges the constraints for instances U1 and U3 into a single context description and merges the constraints for instances U2 and U4 into another context description.

```
characterize_context -block BLKA -instances {U1 U3} -name BLKAodd
characterize_context -block BLKA -instances {U2 U4} -name BLKAeven
update_timing
write_context {U1 U3 U2 U4} -format gbc BLKA_context_dir
```

See Also

- remove_context
- report_context
- write_context

---

check_activity

Check the VCD and FSDB file and reports any unannotated, X, Z, or unconnected states of the input signals.

Syntax

bool *check_activity*

```
[-trace_fanin]
[-show_pins]
pin_list | net_list
```

Data Types

pin_list | *net_list* list

Arguments

`-trace_fanin`

Check the fanin of the pins specified in the pin list for X or Z states.

`-show_pins`

report the pins specified in the net list for X or Z states.

pin_list | net_list

List of pins or nets on which the activity checker needs to be performed.

Description

Use the *check_activity* command to check if a pin is in X, Z, unannotated, or unconnected states in the VCD/FSDB file. The command can only be used in the time-based power analysis. This command updates the "activity_checker_status" attribute on the pin.

If the initial state of a pin is not annotated, the pin is simply reported as unannotated.

All constant pins (connected to constant logic including TieHI and TieLO cells) are reported as Pass.

Similarly, internal pins of a cell are also marked as Pass. For example, in case of an ICG cell, the output pin of the latch is directly connected to an AND gate inside the ICG hierarchical cell. This internal pin is marked as Pass by the activity checker.

Any unconnected pin which does not have a driver is marked as unconnected by the checker.

If a pin has X value in time (0, 30), then switches to Z in time (30, 60), the initial state of the pin is always used and reported in summary as X. The final Z value is not used for reporting.

Also, activity checker only checks for events in the time interval specified during "read_vcd" command. If VCD/FDSB is partially read, then activity checker can only report status for events during that period.

So, for example, when you read events in (0, 30) time interval using the "read_vcd" command. The pin status is initialized as 'X' in VCD. Therefore, the activity checker only knows about the pin status and reports it as 'X' for the period.

It might be possible that the pin state later changes to Z at 40 ns and remains at Z until end of VCD. After reading events for period "0 - 30 ns", it is not possible to know if the pin state ever changes in the rest of VCD/FSDB. So, activity checker reports the current output of pin as 'X' from (0, unknown) after reading events from "0 - 30 ns". This simply means the pin never changed state during the time interval specified in "read_vcd".

Examples

The following examples show how to use activity checker in PrimePower.

The following example reads a VCD file mini.vcd from the disk and uses the first 30 time unit of the events in the file. If the time unit is ns, this command reads first 30 ns of events from VCD.

```
pwr_shell> read_vcd $test_dir/mini.vcd  -strip_path testmini/mini -time
 {0 30} -rtl
```

```
============================================================================
Summary:
Total number of nets = 9
Number of annotated nets = 8 (88.89%)
Total number of leaf cells = 4
Number of fully annotated leaf cells = 4 (100.00%)
============================================================================

Information: Total number of synthesis invariant points = 6 ,
annotated synthesis invariant points = 6, annotation_ratio = 100.00%
1
```

The following example shows how to use the activity checker after reading "mini.vcd" from the previous command. The summmary report shows one pin (q) in X state, and one pin in Z state (in2), while the rest of the pins are in Pass state during (0 - 30 ns) time period.

The detailed report shows that "in2" pin becomes Z at 10 ns and then stays at Z until 30 ns, while "q" pin started with X state and never changed state during (0, 30) time period.

```
pwr_shell> check_activity
[get_pins -of_object [get_cells * -hier -filter "is_hierarchical ==
 false"]]

============================================================================
=======
State       State        State         State         Pass
Unconnected   X            Z           Unannotated     0/1
----------------------------------------------------------------------------
-------
   0          1             1             0            11
( 0.00%)   ( 7.69%)      ( 7.69%)      ( 0.00%)      (84.62%)
============================================================================
=======

============================================================================
State     Time Interval         Net
----------------------------------------------------------------------------
  X     (0 unknown)               q
  Z     (10 30)                   in2
============================================================================
1
```

The following example reads a VCD file mini.vcd from the disk and uses the full 100 ns of the events in the file.

```
pwr_shell> read_vcd $test_dir/mini.vcd  -strip_path testmini/mini -rtl

============================================================================
Summary:
Total number of nets = 9
Number of annotated nets = 8 (88.89%)
Total number of leaf cells = 4
```

```
Number of fully annotated leaf cells = 4 (100.00%)
============================================================================

Information: Total number of synthesis invariant points = 6, annotated
 synthesis
invariant points = 6, annotation_ratio = 100.00%
1
```

The following example shows the activity checker after reading "mini.vcd" from previous command.

The summary report shows one pin (q) in X state, two pins in Z state (in2 and clk).

The detailed report shows that "q" pin remained in X state during (0 50) time interval. Here, "in2" pin was in Z state during (0-30ns) time period and then became X at 40 time unit. The in2 pin state did not change until end of all events in VCD file (40 unknown).

Similarly, "clk" pin was in Z state from (100, 150) and (200, 230) time units.

```
pwr_shell> check_activity
[get_pins -of_object [get_cells * -hier -filter "is_hierarchical ==
 false"]]

============================================================================
=======
State       State        State         State         Pass
Unconnected   X            Z           Unannotated     0/1
----------------------------------------------------------------------------
-------
   0          1             2             0            10
( 0.00%)   ( 7.69%)      (15.38%)      ( 0.00%)      (76.92%)
============================================================================
=======

============================================================================
State     Time Interval         Net
----------------------------------------------------------------------------
  X     (0 50)                    q
  Z     (10 30)                   in2
  X     (40 unknown)              in2
  Z     (100 150)                 clk
  Z     (200 210)                 clk
============================================================================
1

pwr_shell> check_activity -show_pins [get_nets -of_object [get_cells *
 -hier -filter "is_hierarchical == false"]]
============================================================================
=======
State       State        State         State         Pass
Unconnected   X            Z           Unannotated     0/1
```

```
------------------------------------------------------------------------
-------
   0          1             2             0             7
( 0.00%)  (10.00%)      (20.00%)      ( 0.00%)      (70.00%)
========================================================================
=======

========================================================================
State    Time Interval           Net             Pins
------------------------------------------------------------------------
  Z      (10 30)                   in2            U2/B
  X      (40 unknown)              in2            U2/B
  X      (0 50)                    q               q
========================================================================
1
```

See Also

- read_vcd
- set_switching_activity
- reset_switching_activity
- report_power

---

check_constraints

Checks constraints of the current design and reports potential problems.

Syntax

int *check_constraints*

```
[-verbose]
```

Arguments

`-verbose`

Print verbose report of constraint errors.

Description

Invokes constraint consistency to check constraints of the current design for potential problems.

Feedback

Examples

The following example shows a constraint checking report:

```
pt_shell> check_constraints -verbose

****************************************
Report : check_constraints
 ...
****************************************

Scenario Violations  Count Waived Description
--------------------------------------------------------------------------------
-----
Design: counter
<Global Violations>     1     0   Scenario independent violations
  error                 1     0
    UNT_0002            1     0   Library 'library' has incomplete units
 defined.
      1 of 1                0   Library
 '/u/zinmgr/test_data/ext_libs/test10k.db:test10k' has
                                     incomplete units defined.
default                 8     0   Default scenario violations
  warning               8     0
    EXD_0001            4     0   Input/inout port 'port' has no input
 delay specified
      1 of 4                0   Input/inout port 'A' has no input delay
 specified
      2 of 4                0   Input/inout port 'B' has no input delay
 specified
      3 of 4                0   Input/inout port 'C' has no input delay
 specified
      4 of 4                0   Input/inout port 'D' has no input delay
 specified
    EXD_0003            4     0   Output/inout port 'port' has no
 clock-related output delay
                                     specified
      1 of 4                0   Output/inout port 'QA' has no
 clock-related output delay specified
      2 of 4                0   Output/inout port 'QB' has no
 clock-related output delay specified
      3 of 4                0   Output/inout port 'QC' has no
 clock-related output delay specified
      4 of 4                0   Output/inout port 'QD' has no
 clock-related output delay specified
--------------------------------------------------------------------------------
-----
Total Error Messages    1     0
Total Warning Messages  8     0
Total Info Messages     0     0
```

Feedback

See Also

- check_timing
- report_case_analysis

---

check_eco

Reads in the LEF/DEF or IC Compiler II physical library and design information and checks the data for correctness and consistency with the logical netlist.

Syntax

```
status check_eco
```

Arguments

None

Description

The *check_eco* command reads physical library and design files in either LEF/DEF or IC Compiler II format as previously specified by the *set_eco_options* command, and checks the data for correctness and consistency with the logical netlist previously read by the *read_verilog* command.

The *check_eco* command reports any errors, warnings, or missing LEF macros in the log file specified by the *set_eco_options* command. Be sure to resolve any reported errors. See the man pages for the reported error codes as needed. After you fix the errors, rerun *check_eco* to repeat the checking process.

If physical LEF or DEF files are missing, you can run *reset_eco_options* to reset the previous *set_eco_options* settings. Then you can specify all physical information files with *set_eco_options* and rerun *check_eco*.

A successful *check_eco* run loads the physical information in PrimeTime. Subsequent physically aware ECO commands (*fix_eco_timing*, *fix_eco_drc*, and *fix_eco_power*) operate on the physical information in memory; they do not reload the physical information.

To view the chip layout in the GUI, choose Window > Layout Window. This opens a new layout window showing any ECO changes. From this window, you can perform further ECO operations manually using the following commands:

```
ECO > Insert Buffer
ECO > Size Cell
ECO > Remove Buffer
ECO > Create Placement Blockage
```

You can also display color-coded maps using the following commands:

```
View > Map > Cell Density
View > Map > Pin Density
View > Map > Cell Power Density
```

PrimeECO flow

In the context of the PrimeECO physical signoff closure flow, the *check_eco* command performs designated tasks. In that flow, you must specify the following information:

```
set_eco_options \\
  -physical_icc2_lib my_NDM \\
  -physical_icc2_blocks CPU_after_route

set_implement_options \\
  -icc2_exec icc2_shell \\
  -starrc_exec StarXtract
```

PrimeECO enables incremental ECO operation by keeping the PrimeTime-ECO, IC Compiler II, and StarRC tools synchronized. For that purpose, the *check_eco* command does the following:

- Invokes IC Compiler II to load a working view of the design block specified by the *set_eco_options -physical_icc2_block* option, and starts recording ECO changes
- Reads physical data to enable physically aware optimization by *fix_eco_timing*, *fix_eco_drc*, and *fix_eco_power*
- If StarRC incremental extraction is enabled, runs full extraction with StarRC to establish a baseline for subsequent incremental extraction

For more details about the PrimeECO flow, see the man page for the *implement_eco* command.

Examples

The following script specifies the physical library and design data in LEF/DEF format. The *check_eco* command loads the physical data and checks it for correctness and consistency with the logical netlist. The *gui_start* command opens the GUI main window, from which you can use the Window > Layout Window command to display any physical ECO changes.

```
set_eco_options \\
  -physical_tech_lib_path LEF_tech_file_list \\
  -physical_lib_path LEF_lib_file_list \\
  -physical_design_path DEF_design_file_list
check_eco
gui_start
```

Feedback

The following script specifies the physical library and design data in IC Compiler II format, loads and checks the physical data, and starts the GUI.

```
set_eco_options \\
  -physical_icc2_lib icc2_lib_directory_path \\
  -physical_icc2_blocks icc2_block_name_list
check_eco
gui_start
```

See Also

- implement_eco
- read_eco_changes
- report_eco_options
- reset_eco_options
- reset_implement_options
- set_eco_options
- set_implement_options

---

check_level_shifters

Alias for the *check_timing -override_defaults {signal level}* command.

Syntax

string *check_level_shifters*

```
[-verbose]
```

Arguments

`-verbose`

Prints a detailed report of all possible violations.

Description

In PrimeTime, the *check_level_shifters* command is an alias for:

```
check_timing -override_defaults {signal_level}
```

See Also

- check_timing

Feedback

check_noise

Checks that the necessary data is available to run the *update_noise* command.

Syntax

int *check_noise*

```
[-verbose]
[-beyond_rail]
[-nosplit]
[-include noise_driver | noise_immunity]
```

Arguments

`-verbose`

Enables verbose mode showing detailed information and pin names; shows cell and pin names that have no model or invalid models.

`-beyond_rail`

Enables beyond_rail noise analysis check. By default, *check_noise* performs only within_rail analysis.

`-nosplit`

Prevents line-splitting and facilitates writing software to extract information from the report output, because most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`-include noise_driver | noise_immunity`

Specifies one of the following types of checking: *noise_driver* and *noise_immunity* (the default).

Description

It is possible to have invalid noise models in a library and run noise analysis without detecting them. This might result in inaccurate results. If the design is large, and it takes a long time to finish the noise analysis, it also causes longer turnaround time. Moreover, it is very important that all pins in the design have correct noise immunity information. Otherwise, when the noise analysis is over, violations are not reported even if noise bumps are large enough to create violations.

To validate the correctness of a design with respect to the noise analysis, run the *check_noise* command before performing noise analysis. It checks any invalid noise model, any pin without a model from the library, and the design. It checks if all pins in the design are 'noise constrained', for example, so their noise immunity can be calculated.

The command might produce different results before and after *update_timing*. For example, case analysis can disable noise models, and pins can be unconstrained after *update_timing*. It is recommended that you run the *check_noise* command after *update_timing* to consider design context.

Examples

The following example shows the noise immunity check.

```
pt_shell> check_noise

  Noise immunity type            above_low  below_high
  ------------------------------------------------------
  user hyperbolic curve                  8           8
  user margin                            1           1
  library immunity table                 0           0
  library hyperbolic curve               0           0
  library CCS noise immunity          3245        3245
  none                                   0           0
```

The following example shows the noise driver check.

```
pt_shell> check_noise -include "noise_driver"

Noise driver check:

  Noise driver type              above_low    below_high
  ------------------------------------------------------
  CCS noise                            520         520
  library IV curve                     121         121
  library resistance                     0           0
  equivalent library pin                 1           1
  user resistance                        1           1
  none                                   1           1
```

The verbose mode shows which pin is not constrained.

```
pt_shell> check_noise -include "noise_driver" -verbose

Noise driver check:

  library pin name                             Region           Status

 ------------------------------------------------------------------------
----
  top/inv1/Z                                   above_low        None

  Noise driver type              above_low    below_high
  ------------------------------------------------------
  CCS noise                            520         520
  library IV curve                     121         121
  library resistance                     0           0
  equivalent library pin                 1           1
```

```
  user resistance                          1           1
  none                                     1           1
```

See Also

- report_noise
- set_noise_parameters
- update_noise

---

check_power

Shows possible power problems for design and specific cell object_list.

Syntax

string *check_power*

```
[-verbose]
[-significant_digits digits]
[-override_defaults check_list]
[-include check_list]
[-exclude check_list]
[-max_fanout fanout]
[-max_slew   slew]
[-max_cap    cap]
[-max_delay  delay]
[object_list]
```

Data Types

```
digits        integer
check_list    list
fanout        integer
slew          float
cap           float
delay         float
```

Arguments

`-verbose`

Shows detailed information about potential problems.

`-significant_digits digits`

Specifies the number of digits of precision to be displayed by warnings that show floating point numbers. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default value is 2. Use this option if you want to override the default.

`-override_defaults` *check_list*

Overrides the checks in *power_check_defaults* variable with those listed in *check_list*. The allowed checks are:

```
no_arrival_window
no_base_clock
out_of_table_range
missing_table
missing_function
```

`-include` *check_list*

Adds the checks listed in *check_list* to the checks in *power_check_defaults*.

`-exclude` *check_list*

Subtracts the checks listed in *check_list* from the checks in *power_check_defaults*.

`-max_fanout` *fanout*

Specifies maximum fan-out limit for checking fan-out for each of driver nets.

`-max_slew` *slew*

Specifies maximum slew limit in library time units for checking slew for each of the input pins.

`-max_cap` *cap*

Specifies maximum capacitance limit in library cap units for checking capacitance for each of the driver nets.

`-max_delay` *delay*

Specifies maximum delay limit in library time units for checking delay for each of the arcs.

`object_list`

Accepts a collection of cell or library cell objects.

Description

The check_power command checks structure of the design for potential power violations. This command is used to identify possible power calculation problems before updating power or before generating power reports.

This command also prints which checks it performs. If a check reveals a violation, then the command also prints a message about the violation. By default, the message contains a summary of the violation. To get more information about violations, use the *-verbose* option.

This command without any options performs the checks defined by the *power_check_defaults* variable. To change the value of this variable, either redefine it or use the *-override_defaults* option. If the *-override_defaults* option is not used, the final list of checks to be performed is the checks in *power_check_defaults* plus the list of checks given by the *-include* option minus the list of checks given by the *-exclude* option. If the *-override_defaults* option is used with a check_list, the final list of checks to be performed is the checks in the check_list given by the option plus the list of checks given by the *-include* option minus the list of checks given by the *-exclude* option.

The alphabetically ordered list below shows the meaning of each check:

missing_function

Checks if library cells have function statements. The function statements are used during toggle propagation and event simulation.

missing_table

Checks if library cells have leakage power tables and dynamic power tables.

no_arrival_window

Checks whether pins have arrival window or not. This check helps to generate accurate cycle-based average waveform, zero delay VCD power analysis, and RTL VCD power analysis. In order to save the arrival windows user should set the variable timing_save_pin_arrival_and_slack to true before update_timing/ update_power. Users seeing this violation are directed to check if timing is correct by using the *check_timing* command.

no_base_clock

Checks whether nets have power base clock or not. During updating power, the fastest clock is used as power base clock for these nets that do not have power base clock. However, this exposes some potential accuracy problem. Users seeing this violation are directed to check if timing is correct by using the *check_timing* command.

out_of_table_range

Checks if there are out-of-range ramp or load violations when looking up power tables for each cells. The out of range values together with cell name, pin name, and library cell name and its range is displayed if the "-verbose" option is used.

Examples

The following example sets maximum slews as 0.4, maximum capacitance as 0.25, maximum delay as 0.2 and maximum fan-out as 200 and shows detailed information on potential problems in the design.

```
pt_shell> check_power -max_slew 0.4 -max_cap 0.25 -max_delay 0.2
 -max_fanout 200 -verbose
```

Feedback

See Also

- power_check_defaults
- check_timing
- report_constraint

---

check_routes

Verifies and reports routing design rule checking (DRC) violations, net opens, antenna rule violations, voltage-area rule violations, and via conversion rates.

Syntax

```
status check_routes
```

Arguments

None

Description

This command verifies, checks, and reports routing statistics for the current design block. Only signal routes are included in the check; PG routes are ignored.

The PrimeECO *check_routes* command works like the *check_routes* command in IC Compiler II with the default command options. For details, see the IC Compiler II man page for the *check_routes* command.

Examples

The following example reports physical routing DRC violations for all nets in the block.

```
prompt> check_routes

  ...

DRC-SUMMARY:
  @@@@@@@ TOTAL VIOLATIONS = 94
  Diff net spacing : 42
  Edge-line via spacing : 1
  End of line spacing : 3
  Fat wire via keepout enclosure : 48
```

See Also

- check_eco
- implement_eco

Feedback

- report_implement_options
- reset_implement_options
- save_block
- set_eco_options
- set_implement_options
- write_implement_changes

---

check_rtl_power

Checks for potential redundancies or issues in RTL design.

Syntax

status *check_rtl_power*

```
[-check_type rtl_check_list]
[-check_group group_list]
[-topN number]
[-level number]
[-csv csv_file]
[-root hier_cell_list]
[-hier hier_cell_list]
[-exclude_hier hier_cell_list]
[-clock clock_list]
[-file file_list]
[-verbose]
[-parent_reference]
[object_list]
```

Data Types

```
rtl_check_list    list
group_list        list
number            number
csv_file          string
hier_cell_list    list
clock_list    list
file_list    list
```

Arguments

`-check_type rtl_check_list`

Specifies the rtl checks to be performed on the design.

Check group type Lint checks

ClockGating CICG, FICG, RICG, SCECG, UCGREG, UCNREG, IFRREG, INSREG, BTHREG, CCGREG and XORREG

ConstantInput ENHREG and ENLREG

SeqOpt DUPREG, CNDREG, CNRREG, UNOREG, CNDLCH, CNELCH and UNOLCH

Memory MCCG, UCGMEM, CGPMEM, ENHMEM, ENLMEM and ODCMEM

Glitch OPRMUX and OPRDPT

By default all checks are performed.

`-check_group` *`group_list`*

Report all the lint checks belonging to the specified group list. Allowed values are : ClockGating, ConstantInput, SeqOpt, Memory, Glitch.

`-topN` *`number`*

When used with verbose mode, print only this number of violations for each tag. Useful to limit the file size for designs with a large violation count.

`-level` *`number`*

Specifies the depth (n) for reporting registers with clock enable depth > n or operators with input depth difference > n. By default depth > 5 are reported.

`-csv` *`csv_file`*

This option specifies that the report is written out in a comma-separated value format. The *csv_file* specifies the csv file name for the report.

`-root` *`hier_cell_list`*

Report lint checks only for the cells inside the specified hierarchical cells.

`-hier` *`hier_cell_list`*

Report lint checks only for the cells inside the specified hierarchical cells.

`-exclude_hier` *`hier_cell_list`*

Skip lint checks for the cells inside the specified hierarchical cells.

`-clock` *`clock_list`*

Report lint checks only for the cells that belong to specified clock domains.

`-file` *`file_list`*

Report lint checks only for the cells originating from specified rtl files.

`-verbose`

Specifies to display detailed report

`-parent_reference`

Report the file name and line number for the parent hierarchy along with the file name and line number of the cell. This option can be provided only if the -verbose option is enabled.

Description

The *check_rtl_power* command flags off the potential redundancies and issues in RTL design. It generates detailed instance level report when *-verbose* option is given.

User can specify one or more checks through *-check_type* option. The following types of check are supported :

- *CICG* - Constant ICG inputs i.e at least one of EN or CLK pins of ICG is tied to high/low. Specify using *-check_type constant_icg*.
- *FICG* - Floating ICG i.e ICG output is not driving any cell. Specify using *-check_type floating_icg*.
- *RICG* - Redundant cascaded ICGs. Specify using *-check_type redundant_icg*.
- *SCECG* - ICGs having common enable and clock. Specify using *-check_type same_clock_enable_cg*.
- *DUPREG* - Duplicate registers i.e. registers having common data and clock. Specify using *-check_type duplicate_reg*.
- *UCGREG* - Ungated register count. Specify using *-check_type UCGREG*.
- *UCNREG* - Untraced register count. Specify using *-check_type UCNREG*.
- *IFRREG* - Inferred only gated register count. Specify using *-check_type IFRREG*.
- *INSREG* - Instantiated only gated register count. Specify using *-check_type INSREG*.
- *ENHREG* - Registers with tied-high clock enables. Specify using *-check_type ENHREG*.
- *ENLREG* - Registers with tied-low clock enables. Specify using *-check_type ENLREG*.
- *CCGREG* - Registers with clock enable depth > n. Specify using *-check_type CCGREG -level n*. Default value of n is 5.
- *XORREG* - Registers with XOR based self gating. Specify using *-check_type XORREG*.
- *UCGMEM* - Ungated memory clock ports. Specify using *-check_type UCGMEM*.

- *CGPMEM* - Clock-gated memory clock ports. Specify using *-check_type CGPMEM*.
- *ENHMEM* - Memories with tied-HIGH enables. Specify using *-check_type ENHMEM*.
- *ENLMEM* - Memories with tied-LOW enables. Specify using *-check_type ENLMEM*.
- *CNDREG* - Number of registers optimized due to DATA constant. Specify using *-check_type CNDREG*.
- *CNRREG* - Number of registers optimized due to RESET constant. Specify using *-check_type CNRREG*.
- *UNOREG* - Number of registers optimized due to OUTPUT unused. Specify using *-check_type UNOREG*.
- *CNDLCH* - Number of latches optimized due to DATA constant. Specify using *-check_type CNDLCH*.
- *CNRLCH* - Number of latches optimized due to RESET constant. Specify using *-check_type CNRLCH*.
- *UNOLCH* - Number of latches optimized due to OUTPUT unused. Specify using *-check_type UNOLCH*.
- *BTHREG* - Inferred & Instantiated gated register count. Specify using *-check_type BTHREG*.
- *ODCMEM* - Memories with ODC Condition. Specify using *-check_type ODCMEM*.
- *OPRMUX* - Operators with mux driving the input. Specify using *-check_type OPRMUX*.
- *OPRDPT* - Operators with input depth difference > n. Specify using *-check_type OPRDPT -level n*. Default value of n is 5. -

User can specify one or more check group type through *-check_group* option to report all lint checks belonging to it. The following types of check groups are supported (along with the lint checks) :

- *ClockGating* - Report for Lint Checks : CICG, FICG, RICG, SCECG, UCNREG, UCGREG, IFRREG, INSREG, BTHREG, CCGREG, XORREG.
- *ConstantInput* - Report for Lint Checks : ENHREG, ENLREG.
- *SeqOpt* - Report for Lint Checks : DUPREG, CNDREG, CNRREG, UNOREG, CNDLCH, CNELCH, UNOLCH.
- *Memory* - Report for Lint Checks : MCCG, UCGMEM, CGPMEM, ENHMEM, ENLMEM, ODCMEM.
- *Glitch* - Report for Lint Checks : OPRMUX, OPRDPT.

Feedback

Examples

The following examples display usage of check_rtl_power command.

```
pt_shell> check_rtl_power
****************************************
Report : check_rtl_power
Design : top
****************************************


--------------------------------------------------------------------------------
-------------------------------------
Family         Check          Check info
       Count          Percentage
--------------------------------------------------------------------------------
-------------------------------------
ClockGating    INSREG         Instatiated only gated register
       4              22.22
ClockGating    BTHREG         Inferred and Instantiated gated register
       4              22.22
ClockGating    IFRREG         Inferred only gated register
       4              22.22
ClockGating    UCGREG         Ungated register
       6              33.33
ClockGating    UCNREG         Untraced register
       4              22.22
--------------------------------------------------------------------------------
-------------------------------------

pt_shell> check_rtl_power -check_type "constant_icg floating_icg"
 -verbose
****************************************
Report : check_rtl_power
Design : test
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
Family         Check          Check info
                RTL Name
 File_Name:Line_Number
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
ClockGating    CICG           Constant ICG
       cg1                                                  rtl.v:12

       cg2                                                  rtl.v:10
ClockGating    FICG           Floating ICG
       cg2                                                  rtl.v:10
```

Feedback

```
------------------------------------------------------------------------
------------------------------------------------------------------------
--------------------------------
1
```

check_script

Statically check a script using the a static Tcl syntax checker based on TclPro Procheck and the Synopsys checking extensions for this tool.

Syntax

string *snpsTclPro::check_script* [-no_tclchecker_warning]

[-quiet] [-onepass] [-verbose] [-suppress *messageID*] [-application *appName*] [-W1] [-W2] [-W3] [-Wall] *filePattern*

```
string messageID
string appName
string filePattern
```

Arguments

`-no_tclchecker_warning`

Don't warn that TclDevKit cannot be found

`-quiet`

prints minimal error information

`-onepass`

perform a single pass without checking proc args

`-verbose`

prints summary information

`-suppress messageIDs`

prevent given messageIDs from being printed

`-application appName`

Check script against application other than this one

`-W1`

print only error messages

`-W2`

print only error and usage warnings

`-W3`

print all errors and warnings

`-Wall`

print all types of messages (same as W3)

`filePattern`

file(s) to be checked

USING THE PACKAGE

The *check_script* command is in the snpsTclPro Tcl package, and all of the commands in this package are defined in the namespace snpsTclPro. The way to use this command is to load the package, which will import the commands it exports into the global namespace. This is shown below.

```
shell> package require snpsTclPro
1.0
```

These commands can be added to the .synopsys setup file for the system, user, or current directory, or the commands can be typed when the package is needed.

Description

The *check_script* command simplifies running the TclDevKit tclchecker program. This application is available for purchase from ActiveState (http://www.activestate.com). If the TclDevKit is not available then there is a limited Synopsys Tcl syntax checker that is used by this command. The Synopsys version of this code does not understand Tcl8.4 constructs, and some other checking is limited.

These checkers will check builtin Tcl commands, as well as the Synopsys extension commands which are found in the script. *check_script* will locate the Synopsys checking extensions for the application and will then invoke the checker. The limited checker does not require any external license. It is part of the Synopsys installation.

The *check_script* command will raise a Tcl error if the checker detects errors in the scripts being checked.

The *check_script* command is provided as a convenience to streamline the use of the syntax checker within Synopsys applications. The checker can also be called directly on scripts, without requiring a license for the Synopsys application whose script is being checked. This checker script is available in the auxx/tcllib/bin/check_script in the installation directory of your application. When run standalone the -application option must be specified.

Synopsys Checker Extensions

The Synopsys extensions to the TclPro checker define a number of new warnings and errors. These messages are listed below, along with a short explanation of them.

Feedback

Error *MisVal*

An option to a command which requires a value is missing its value.

Error *MisReq*

A required option or argument was not specified.

Error *Excl*

Two options which are mutually exclusive were both specified.

Error *ExtraPos*

An extra positional argument was specified to a command.

Error *BadRange*

The value specified for a given argument was outside of the legal range.

Error *UnkCmd*

The command specified is not known by the application.

Error *UnkOpt*

The specified option is not legal for the command.

Error *AmbOpt*

The option specified is not complete and matches 2 or more options for the command.

Warning *NotLit*

This warning indicates that the argument specified to an option was not a literal and therefore could not be checked. This message is suppressed by default. It can be enabled by setting the environment variable SNPS_TCL_NOLITERALWAN to true.

Warning *DupIgn*

An option was specified multiple times to the command, and the first value specified will be the one used. The other values will be ignored.

Warning *DupOver*

An option was specified multiple times to the command, and the last value specified will be the one used. The other values will be ignored.

Examples

```
shell> check_script myscript.tcl

Loading snps_tcl.pcx...
Loading coreConsultant.pcx...
```

Feedback

```
scanning: /u/user1/myscript.tcl
checking: /u/user1/myscript.tcl
test.tcl:3 (warnVarRef) variable reference used where variable name
 expected
set $a $b
    ^
test.tcl:5 (SnpsE-MisReq) Missing required positional options for
 foreach_in_collection: body
foreach_in_collection x {
}
^
test.tcl:9 (SnpsE-BadRange) Value -1 for 'index_collection index' must be
 >= 0
index_collection $a -1
child process exited abnormally
Error: Errors found in script.
        Use error_info for more info. (CMD-013)

shell> check_script -Wall anotherScript.tcl

Loading snps_tcl.pcx...
scanning: /u/user1/anotherScript.tcl
checking: /u/user1/anotherScript.tcl
```

CAVEATS

The Synopsys extension messages cannot be suppressed.

Aliases and abbreviated commands will be flagged as undefined procedures.

See Also

- debug_script

---

check_signal_em

performs signal nets electromigration analysis

Syntax

status *check_signal_em*

```
[net_list]
-skip_filter
```

Data Types

```
net_list      list
```

Feedback

Arguments

net_list

The nets list to perform signal electromigration analysis. If the list is empty, all signal nets are checked.

-skip_filter

Skip fast filtering step. The command has two major steps. The first one is faster filtering step. This step identifies signal nets which may have EM violations, then pass these nets to step 2. The second step performs detailed EM analysis and prepares analysis result for *report_signal_em_violation* command. This option skips the faster filtering step, then all signal nets will perform detail EM analysis.

Description

The command performs electromigration analysis on given signal nets. Electromigration analysis is used to determine whether the current density at a location in the circuit exceeds the defined electromigration rule limits. The current density limits are intended to control the risk that current-carrying geometries is degraded by excessive current flow. High current densities damage the wires and damaged current-carrying wires can ultimately lead to shorts and opens within the metal, causing circuit failure.

Note that you must set variable *power_enable_signal_em_analysis* to TRUE before *check_signal_em* and *read_parasitics* command.

The foundries set the current density limit rules of their related metallurgy to ensure reliability. Before you run this command, you must load electromigration rule file by *read_signal_em_rules* command. See the *read_signal_em_rules* man page for more details.

The command gets signal net geometries from tail comments of parasitics netlist written by StarRC. Before you run this command, you must load signal net parasitics by *read_parasitics* command. It is recommended to use *-format SPEF* of *read_parasitics*, since it has better tail comment support.

To output SPEF parasitics netlist for signal electromigration analysis, the recommend StarRC options are listed below:

```
EXTRACTION: RC
REDUCTION: NO
NETS: *
NETLIST_CONNECT_OPENS: !*
NETLIST_NODE_SECTION: YES
NETLIST_TAIL_COMMENTS: YES
EXTRA_GEOMETRY_INFO: NODE RES
SHORT_PINS: NO
KEEP_VIA_NODES: YES
EM_EXTARCTION: YES
```

Feedback

You can use *set_signal_em_analysis_options* command to set options of signal electromigration analysis. Refer to *set_signal_em_analysis_options* man page for option details.

You can use *report_signal_em* command to report signal electromigration violations. Refer to *report_signal_em* man page for more details.

Examples

The following command invokes signal em analysis for all signal nets.

```
pt_shell> check_signal_em
```

The following command invokes signal electromigration analysis for net net001.

```
pt_shell> check_signal_em {net001}
```

See Also

- read_signal_em_rules
- set_signal_em_analysis_options

---

check_timing

Reports possible timing problems in the design.

Syntax

string *check_timing*

```
[-verbose]
[-significant_digits digits]
[-ms_min_separation delta]
[-override_defaults check_list]
[-include check_list]
[-exclude check_list]
[-sms_scenarios sms_scenarios_list]
```

Data Types

digits	int
delta	float
check_list	list
sms_scenarios_list	collection

Arguments

`-verbose`

Shows detailed information about potential problems.

`-significant_digits` *`digits`*

Specifies the number of digits of precision to be displayed by warnings that show floating point numbers. Allowed values are 0-13. The default is determined by the *report_default_significant_digits* variable, which has a default of 2. Use this option if you want to override the default.

`-ms_min_separation` *`delta`*

Minimum separation value between master and slave clocks. The default minimum separation is 0.0.

`-override_defaults` *`check_list`*

Overrides the checks in the *timing_check_defaults* variable using the *check_list* option. For default information, see the *timing_check_defaults* man page.

`-include` *`check_list`*

Adds the checks specified in the *check_list* to the checks in *timing_check_defaults* variable.

`-exclude` *`check_list`*

Subtracts the checks specified in the *check_list* from the checks in *timing_check_defaults* variable.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only timing problems related to the specified SMS scenarios collection are reported. This option is supported only with *operating_conditions* check. This collection is created by *get_sms_scenarios* command.

Description

Checks the assertions and structure of the design for potential timing violations. This command is used to identify possible problems before generating timing or constraint reports.

This command also prints which checks it performs. If a check reveals a violation, the command also prints a message about the violation. By default, the message contains a summary of the violation. To get more information about violations, use the *-verbose* option.

This command without any options performs the checks defined by the *timing_check_defaults* variable. To change the value of this variable, either redefine it or use the *-override_defaults* option. If the *-override_defaults* option is not used, the final list of checks to be performed is the checks in the *timing_check_defaults* variable plus the list of checks given by the *-include* option minus the list of checks given by the *-exclude* option. If the *-override_defaults* option is used with a *check_list*, the final list of checks to

be performed is the checks in the *check_list* given by the option plus the list of checks given by the *-include* argument minus the list of checks given by the *-exclude* argument.

You can specify the following checks in the *check_list* argument:

- *clock_crossing* - Checks clock interactions when there are multiple clock domains. If a clock launches one or more paths, which are captured by other clocks, it is listed in clock crossing report. If all paths between two clocks are false paths or they are exclusive/asynchronous clocks, the path is marked by *. If only part of paths are set as false paths or exclusive/asynchronous clocks, the path is marked by #.
- *data_check_multiple_clock* - Warns if multiple clocked signals reach a data check register reference pin. The analysis is performed for each of the clocked domain separately.
- *data_check_no_clock* - Warns if no clocked signal reaches a data check register reference pin. In this case, no setup or hold checks are performed on the constrained pin.
- *generated_clocks* - Checks generated clock network. The master must be driven by a clock source. There cannot be loops of generated clocks. For example, the source of generated clock CLK1 cannot be used to generate clock CLK2 if CLK2 also is used to generate CLK1. Also checks for muxes with clock propagated to both input and select pins. It is possible to stop clock signals from propagating through the mux select pin by using: set_sense -stop_propagation[get_pin $select_pin]
- *generic* - Warns about generic (unmapped) cells in the design. The timing of paths through generic cells is inaccurate because generic cells have zero delay.
- *ideal_clocks* - Shows the clocks that are not defined as propagated using the *set_propagated_clock* command. Generally, all clocks should be propagated so that the clock network timing is accurately calculated. Especially, in presence of crosstalk, the delay changes induced by other nets on the clock network are not reflected in the calculated slacks in the design.
- *latch_fanout* - Checks fanout of level-sensitive latches. A warning is issued if a level-sensitive latch fans out to itself. An information message also appears for a latch that fans out to a latch of the same clock (PHI1 to PHI1 path).
- *latency_override* - Warns of clock latency specification conflicts. If clock source latency is defined for both a clock and its port (source pin), the source latency for clock object is ignored. If input_delay is set on clock port, which also has source latency specified, the input_delay is ignored as a source latency. Also warns if more than one clock latency fan out to any latch clock pin.
- *loops* - Warns of nonconstant combinational feedback loops. Combinational feedback loops are not analyzed. If the feedback loop is not broken by *set_disable_timing*, it is automatically broken by disabling one or more timing arcs.

- *constant_loops* - Warns of constant combinational feedback loops. A constant combinational feedback loop is a loop where all the pins in the loop have a case value 0, 1, or S. Combinational feedback loops are not analyzed. If the feedback loop is not broken by *set_disable_timing*, it is automatically broken by disabling one or more timing arcs.
- *ms_separation* - Checks minimum separation of master and slave clock pulses on master/slave latches.
- *no_clock* - Warns if no clock reaches a register clock pin. In this case, no setup or hold checks are performed on data pins related to that clock pin, and the path starting at the clock pin is not relative to a clock.
- *no_driving_cell* - Warns if a port does not have any driving cell. This warning is issued only when the net connected to the port has parasitics. In such case, the accuracy of delay calculation could be affected, as a default strong driver is assumed in absence of driving cell definition. Especially, in presence of crosstalk, a port with no driving cell could act as a strong aggressor which could lead to significant amount of pessimism in the analysis. Also, a port with no driving cell could act as a string victim, which could underestimate the crosstalk effect.
- *no_input_delay* - Warns if no clock related delay specified on an input port, where it propagates to a clocked latch or output port. With *-verbose*, the port name is listed.
- *operating_conditions* - When the design has scaling groups defined by the *define_scaling_lib_group* command, this operating condition check detects improperly set process, voltage, temperature, rail, and operating conditions. Warning are issued for
  - Extrapolation outside the PVT rail operating condition range covered by the scaling group.
  - If *-exact_match_only* is specified in the *define_scaling_lib_group* command, it warns on exact-match failures when the design cell instance operating condition does not match with any of the corner library operating condition.
  - When a design cell instance does not have an associated scaling group, it warns when the operating condition does not match with that of the linked library operating condition.
- *partial_input_delay* - Warns if any port has partially defined input delay. This happens when *set_input_delay -min* is applied on a port to set the min input delay with respect to a clock, however no *set_input_delay -max* is applied to that port to specify the max delay, or vice versa. As a result, some paths starting from the port with partially defined input delay might become unconstrained and some potential violations could be missed.

- *pll_configuration* - Warns if a problem is detected in the configuration of any phase locked loop (PLL) cells. For a PLL to be correctly configured, the PLL output clock should reach the PLL feedback pin. If a PLL is not correctly configured, it does not show any phase adjustment on the PLL output clock.
- *pulse_clock_no_pulse_generator* - Warns if the pulse clock constraints cannot be honored. This could be because pulse clock constraints have been set on clocks that do not drive pulse generators, or the design might not have any pulse generators.
- *pulse_clock_non_pulse_clock_merge* - Warns if pulse clock and normal clock propagate to the same network.
- *retain* - Warns if any of the RETAIN values is larger than its corresponding delay value. The RETAIN values should be less than their corresponding delay values so that they be considered for hold violations. Otherwise, they might be considered for setup violations.
- *signal_level* - Checks that the driver signal level matches the load signal level. The signal levels are determined from the cell specific operating conditions and rail voltages (or UPF), and from the following library attributes: input_signal_level, output_signal_level, input_voltage, output_voltage. The check is performed on all input pins that have input_voltage defined in the timing library. If the driver pin does not have output_voltage defined in the library then the voltage of the rail powering the driver pin is used as the output signal level. With PG libraries the signal level is determined from input/output_signal_level_high (exact voltage in .lib) first, then input/output_signal_level (reference to voltage_map in .lib) if the related_power_pin uses a different_voltage map, and finally from the instance specific voltage of the related power PG pin. If the load pin does not have input_voltage then matching is done based on driver and load rail voltages. Such matching (without library attributes input/output_voltage) is subject to tolerances set by *set_level_shifter_threshold* and *set_level_shifter_strategy* commands. By default the number of mismatches is reported. With *-verbose* the mismatching driver, load and voltage difference are reported.
- *supply_net_voltage* - Checks that each segment of UPF supply nets has voltage assigned to it by *set_voltage* command. It warns about mismatching voltage between port supply voltage against port driving cell voltage. It also warns about mismatching port supply against visible logic. The port supply refers to port supply attributes specified on port objects through the *set_port_attributes* command.
- *unconnected_pg_pins* - Checks that each power and ground pin is connected to a UPF supply net. The connection can be implicit (for example, power domain) or explicit (for example, *connect_supply_net*).
- *unconstrained_endpoints* - Warns about unconstrained timing endpoints. This warning identifies timing endpoints (output ports and register data pins) that are not constrained for maximum delay (setup) checks. To be reported as unconstrained, the endpoints must have a valid timing signal arriving at it. In this context, a valid timing signal

is one that has a valid launch clock and is not a user-designated false path. If the endpoint is a register data pin, it can be constrained by using *create_clock* for the appropriate clock source. You can constrain output ports using the *set_output_delay* or *set_max_delay* commands.

- *no_arrival_endpoints* - Warns about timing endpoints (output ports and register data pins) that have a valid clocked constraint requirement but no valid timing signal to capture. In this context, a valid timing signal is one that has a valid launch clock and is not a user-designated false path.
- *unexpandable_clocks* - Warns if there are sets of clocks for which periods are not expandable with respect to each other. The checking is only done for the related clock domains, such as ones where there is at least one path from one clock domain to the other. This could be because of an incorrectly defined clock period for one or more of the clocks. Another possibility is when asynchronous clocks with unexpandable periods are interacting where they should have been defined in different clock domains.
- *supply_net_without_associated_supply_port* - Warns if there are supply nets without an associated supply port.
- *dangling_startpoints* - Warns about network startpoints which are inactive from a timing point of view (i.e. electrically inactive). This excludes cases where the inactivity is due to *set_disable_timing* or an unclocked register. Examples of such startpoints include: pins without backward arcs, hierarchical pins without fanin nets, input ports without input delays.
- *voltage_level* - Performs the same checks as the *operating_conditions* check, along with the following additional check:
  - If DCALM (*define_cell_alternative_lib_mapping*) is used, it warns of mismatches between the library voltage expected by the DCALM specification and the actual voltage supplied by the *set_voltage* supply specification.

Examples

The following example checks timing of the current design and shows summary information of potential problems. The checks performed are those defined by *timing_check_defaults*.

```
pt_shell> check_timing

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.

Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
Warning: There are 10 endpoints which are not constrained for maximum
 delay.
```

```
Information: Checking 'generic'.

Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'loops'.
Warning: There are 6 timing loops in the design.

Information: Checking 'generated_clocks'.
```

The following example adds the *clock_crossing* check to the list of checks in *timing_check_defaults*.

```
pt_shell> check_timing -include {clock_crossing}

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.

Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
Warning: There are 10 endpoints which are not constrained for maximum
 delay.

Information: Checking 'generic'.

Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'loops'.
Warning: There are 6 timing loops in the design.

Information: Checking 'generated_clocks'.

Information: Checking 'clock_crossing'.

Information: Checking 'pll_configuration'.
```

The following example adds the *clock_crossing* check to the list of checks in *timing_check_defaults* and removes the *loops* check from the same list.

```
pt_shell> check_timing -include {clock_crossing} -exclude {loops}

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.

Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
```

```
Warning: There are 10 endpoints which are not constrained for maximum
 delay.

Information: Checking 'generic'.

Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'generated_clocks'.

Information: Checking 'clock_crossing'.
```

The following example removes the *retain* option from the list of checks in *timing_check_defaults*. Assuming that this check is not included in *timing_check_defaults*, there is nothing to remove. The output in this example is the same as running the command without the *-exclude* option.

```
pt_shell> check_timing -exclude {retain}

Information: Checking 'no_clock'.
Warning: There are 4 register clock pins with no clock.

Information: Checking 'no_input_delay'.

Information: Checking 'unconstrained_endpoints'.
Warning: There are 10 endpoints which are not constrained for maximum
 delay.

Information: Checking 'generic'.

Information: Checking 'latch_fanout'.
Warning: There are 2 level-sensitive latches which fanout to themselves.
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

Information: Checking 'loops'.
Warning: There are 6 timing loops in the design.

Information: Checking 'generated_clocks'.
```

The following example checks only latch fanout and shows detailed information.

```
pt_shell> check_timing -verbose -override_defaults {latch_fanout}
Warning: There are 2 level-sensitive latches which fanout to themselves.

Latch data pin
------------------------------------------------------------
l3/D
l4/D
```

```
Information: There are 2 level-sensitive latches which fanout to latches
 of the same clock.

From Pin            To Pin              Clock         Level
-------------------------------------------------------------------------
------
l1/G                l2/D                C1            positive
l1/G                l3/D                C1            positive
```

The following example gives a list of supply nets without associated supply port and shows detailed information.

```
pt_shell> check_timing -override
 supply_net_without_associated_supply_port
Information: Checking 'supply_net_without_associated_supply_port'.
Warning: There are '4' supply nets without associated supply port.
 (UPF-747)
```

See Also

- create_clock
- report_constraint
- report_timing
- set_clock_groups
- set_case_analysis
- set_disable_timing
- set_false_path
- set_max_delay
- set_output_delay
- create_power_domain
- create_supply_net
- timing_check_defaults
- timing_input_port_default_clock
- report_default_significant_digits

close_drc_error_data

Closes a physical DRC error data collection.

Feedback

Syntax

```
status close_drc_error_data
drc_error_data
[-save]
[-force]
```

Data Types

```
drc_error_data        collection
```

Arguments

drc_error_data

Specifies the collection of physical DRC error data objects to close.

-save

Saves the pending changes in the exported physical DRC error data before closing the error data file.

The *-save* and *-force* options, are mutually exclusive; you can specify only one.

-force

Closes the specified collection of physical DRC error data, discarding any unsaved pending changes.

The *-save* and *-force* options, are mutually exclusive; you can specify only one.

Description

The *close_drc_error_data* command decrements the open count of the specified error data collection.

The tool increments the open count for an error data file each time you run the *open_drc_error_data* or *create_drc_error_data* command for the error data file. The tool decrements the open count for an error data file each time you run the *close_drc_error_data* command. The error data collection is removed from memory when the open count reaches zero.

If the error data has unsaved pending changes, the command fails unless you use the *-save* or *-force* option.

Examples

The following example opens the external physical DRC error data file named my_design_dppinassgn.err, and then closes it.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
```

Feedback

```
prompt> close_drc_error_data $data
1
```

See Also

- open_drc_error_data
- save_drc_error_data
- get_drc_error_data
- create_drc_error_data
- remove_drc_error_data

---

collections

Describes the methodology for creating collections of objects and querying objects in the database.

Description

Synopsys applications build an internal database of objects and attributes applied to them. These databases consist of several classes of objects, including designs, libraries, ports, cells, nets, pins, clocks, and so on. Most commands operate on these objects.

By definition:

A collection is a group of objects exported to the Tcl user interface.

Collections have an internal representation (the objects) and, sometimes, a string representation. The string representation is generally used only for error messages.

Collections represent an ordered sequence of database objects. These collections provide constant time random access to the objects contained in the sequence.

A set of commands to create and manipulate collections is provided as an integral part of the user interface. The collection commands encompass two categories: those that create collections of objects for use by another command, and one that queries objects for viewing. The result of a command that creates a collection is a Tcl object that can be passed along to another command. For a query command, although the visible output looks like a list of objects (a list of object names is displayed), the result is an empty string.

An empty string "" is equivalent to the empty collection, that is, a collection with zero elements.

To illustrate the usage of the common collection commands, the man pages have examples. Most of the examples use PrimeTime as the application. In all cases, the application from which the example is derived is indicated.

Homogeneous and Heterogeneous Collections

A homogeneous collection contains only one type of object. A heterogeneous collection can contain more than one type of object. Commands that accept collections as arguments can accept either type of collection.

Lifetime of a Collection

Collections are active only as long as they are referenced. Typically, a collection is referenced when a variable is set to the result of a command that creates it or when it is passed as an argument to a command or a procedure. For example, in PrimeTime, you can save a collection of design ports by setting a variable to the result of the *get_ports* command:

```
pt_shell> set ports [get_ports *]
```

Next, either of the following two commands deletes the collection referenced by the *ports* variable:

```
pt_shell> unset ports
pt_shell> set ports "value"
```

Collections can be implicitly deleted when they go out of scope. Collections go out of scope for various reasons. An example would be when the parent (or other antecedent) of the objects within the collection is deleted. For example, if our collection of ports is owned by a design, it is implicitly deleted when the design that owns the ports is deleted. When a collection is implicitly deleted, the variable that referenced the collection still holds a string representation of the collection. However, this value is useless because the collection is gone, as illustrated in the following PrimeTime example:

```
pt_shell> current_design
{"TOP"}

pt_shell> set ports [get_ports in*]
{"in0", "in1"}

pt_shell> remove_design TOP
Removing design 'TOP'...

pt_shell> query_objects $ports
Error: No such collection '_sel26' (SEL-001)
```

Iteration

To iterate over the objects in a collection, use the *foreach_in_collection* command. You cannot use the Tcl-supplied *foreach* iterator to iterate over the objects in a collection, because the *foreach* command requires a list, and a collection is not a list. In fact, if you use the *foreach* command on a collection, it destroys the collection.

The arguments of the *foreach_in_collection* command are similar to those of foreach: an iterator variable, the collection over which to iterate, and the script to apply at each iteration. Note that unlike the *foreach* command, the *foreach_in_collection* command does not accept a list of iterator variables.

The following example is an iterative way to perform a query in PrimeTime. For more information, see the *foreach_in_collection* man page.

```
pt_shell> \\
foreach_in_collection s1 $collection {
  echo [get_object_name $s1]
  }
```

Manipulating Collections

A variety of commands are provided to manipulate collections. In some cases, a particular command might not operate on a collection of a specific type. This is application-specific. Consult the man pages from your application.

- *add_to_collection* - This command creates a new collection by adding a list of element names or collections to a base collection. The base collection can be the empty collection. The result is a new collection. In addition, the *add_to_collection* command allows you to remove duplicate objects from the collection by using the *-unique* option.
- *append_to_collection* - This command appends a set of objects (specified by name or collection) to an existing collection. The base collection is passed in through a variable name, and the base collection is modified directly. It is similar in function to the *add_to_collection* command, except that it modifies the collection in place; therefore, it is much faster than the *add_to_collection* command when appending.
- *remove_from_collection* - This command removes a list of element names or collections from an existing collection. The second argument is the specification of the objects to remove and the first argument is the collection to have them removed from. The result of the command is a new collection. For example, in PrimeTime:

  ```
  pt_shell> set dports \\
  [remove_from_collection [all_inputs] CLK]
  {"in1", "in2", "in3"}
  ```

- *compare_collections* - This command verifies that two collections contain the same objects (optionally, in the same order). The result is "0" on success.
- *copy_collection* - This command creates a new collection containing the same objects in the same order as a given collection. Not all collections can be copied.
- *index_collection* - This command extracts a single object from a collection and creates a new collection containing that object. The index operation is done in constant time - it

is independent of the number of elements in the collection, or the specific index. Not all collections can be indexed.

- *sizeof_collection* - This command returns the number of objects in a collection.

Filtering

In some applications, you can filter any collection by using the *filter_collection* command. This command takes a base collection and creates a new collection that includes only those objects that match an expression.

Some applications provide a *-filter* option for their commands that create collections. This allows objects to be filtered out before they are ever included in the collection. Frequently this is more efficient than filtering after the they are included in the collection. The following examples from PrimeTime filters out all leaf cells:

```
pt_shell> filter_collection \\
[get_cells *] "is_hierarchical == true"]
{"i1", "i2"}
pt_shell> get_cells * -filter "is_hierarchical == true"
{"i1", "i2"}
```

The basic form of a filter expression is a series of relations joined together with AND and OR operators. Parentheses are also supported. The basic relation contrasts an attribute name with a value through a relational operator. In the previous example, *is_hierarchical* is the attribute, **==** is the relational operator, and *true* is the value.

The relational operators are

```
==   Equal
!=   Not equal
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
=~   Matches pattern
!~   Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only true or false.

Additionally, existence relations determine if an attribute is defined or not defined, for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
```

These operators apply to any attribute as long as it is valid for the object class. See the appropriate man pages for complete details.

Sorting Collections

In some applications, you can sort a collection by using the *sort_collection* command. It takes a base collection and a list of attributes as sort keys. The result is a copy of the base collection sorted by the given keys. Sorting is ascending, by default, or descending when you specify the *-descending* option. In the following example from PrimeTime, the command sorts the ports by direction and then by full name.

```
pt_shell> sort_collection [get_ports *] \\
{direction full_name}
{"in1", "in2", "out1", "out2"}
```

Implicit Query of Collections

In many applications, commands that create collections implicitly query the collection when the command is used at the command line. Consider the following examples from PrimeTime:

```
pt_shell> set_input_delay 3.0 [get_ports in*]
1
pt_shell> get_ports in*
{"in0", "in1", "in2"}
pt_shell> query_objects -verbose [get_ports in*]
{"port:in0", "port:in1", "port:in2"}
pt_shell> set iports [get_ports in*]
{"in0", "in1", "in2"}
```

In the first example, the *get_ports* command creates a collection of ports that is passed to the *set_input_delay* command. This collection is not the result of the primary command (*set_input_delay*), and as soon as the primary command completes, the collection is destroyed. The second example shows how a command that creates a collection automatically queries the collection when that command is used as a primary command. The third example shows the verbose feature of the *query_objects* command, which is not available with an implicit query. Finally, the fourth example sets the variable iports to the result of the *get_ports* command. Only in the final example does the collection persist to future commands until iports is overwritten, unset, or goes out of scope.

See Also

- add_to_collection
- as_collection

Feedback

compare_collections

Compares the contents of two collections. If the same objects are in both collections, the result is "0" (like string compare). If they are different, the result is nonzero. The order of the objects can optionally be considered.

Syntax

int *compare_collections*

```
[-order_dependent]
collection1
collection2
```

Data Types

collection1	collection
collection2	collection

Arguments

`-order_dependent`

Indicates that the order of the objects is to be considered; that is, the collections are considered to be different if the objects are ordered differently.

collection1

Specifies the base collection for the comparison. The empty string (the empty collection) is a legal value for the *collection1* argument.

collection2

Specifies the collection with which to compare to *collection1*. The empty string (the empty collection) is a legal value for the *collection2* argument.

Description

The *compare_collections* command is used to compare the contents of two collections. By default, the order of the objects does not matter, so that a collection of cells u1 and u2 is the same as a collection of the cells u2 and u1. By using the *-order_dependent* option, the order of the objects is considered.

Either or both of the collections can be the empty string (the empty collection). If two empty collections are compared, the comparison succeeds (that is, *compare_collections* considers them identical), and the result is "0".

Examples

The following example from PrimeTime shows a variety of comparisons. Note that a result of "0" from *compare_collections* indicates success. Any other result indicates failure.

```
pt_shell> compare_collections [get_cells *] [get_cells *]
0
pt_shell> set c1 [get_cells {u1 u2}]
{"u1", "u2"}
pt_shell> set c2 [get_cells {u2 u1}]
{"u2", "u1"}
pt_shell> set c3 [get_cells {u2 u4 u6}]
{"u2", "u4", "u6"}
pt_shell> compare_collections $c1 $c2
0
pt_shell> compare_collections $c1 $c2 -order_dependent
-1
pt_shell> compare_collections $c1 $c3
-1
```

The following example builds on the previous example by showing how empty collections are compared.

```
pt_shell> set c4 ""
pt_shell> compare_collections $c1 $c4
-1
pt_shell> compare_collections $c4 $c4
0
```

See Also

- collections

Feedback

compare_interface_timing

Compares two reports generated by the *write_interface_timing* command.

Syntax

```
int compare_interface_timing
ref_timing_file
cmp_timing_file
[-output file_name]
[-absolute_tolerance atol_list]
[-percent_tolerance ptol_list]
[-capacitance_tolerance ctol_list]
[-ignore ign_list]
[-include cmp_list]
[-quiet]
[-nosplit]
[-sort_by_worst]
[-session session_name]
[-significant_digits digits]
```

Data Types

ref_timing_file	string
cmp_timing_file	string
file_name	string
atol_list	list
ptol_list	list
ctol_list	list
ntol_list	list
ign_list	list
cmp_list	list
session_name	string
digits	int

Arguments

ref_timing_file

Specifies the name of the timing file to be used as the reference in the comparison.

cmp_timing_file

Specifies the name of the timing file to be compared to the reference timing file.

-output *file_name*

Specifies the name of the output file where the results of the comparison information is written. The information written to *file_name* specifies PASS/FAIL status for every parameter compared by the command.

By default, these results are written to the session transcript. All informational, warning, or error messages are still directed to the session transcript, even if this option is specified.

`-absolute_tolerance` *`atol_list`*

Specifies the absolute error tolerance for time data. The default is zero. See below for a description of the tolerance format. If you specify this option together with the *-percent_tolerance* option, both tolerances must be exceeded to cause a FAIL. See the *-capacitance_tolerance* options for other absolute tolerances.

`-percent_tolerance` *`ptol_list`*

This option is similar to the *-absolute_tolerance* option except that it uses the percent relative error instead of an absolute error. The exception is that slack data ignores this option and uses only the absolute tolerance. The default is zero.

See below for a description of the tolerance format. If you specify this option together with either the *-absolute_tolerance*, or *-capacitance_tolerance* option, both tolerances must be exceeded to cause a FAIL.

`-capacitance_tolerance` *`ctol_list`*

Specifies the absolute error tolerance for capacitance data. The default is zero.

See below for a description of the tolerance format. If you specify this option together with the *-percent_tolerance* option, both tolerances must be exceeded to cause a FAIL. *-noise_tolerance* options. For other absolute tolerances, see the *-absolute_tolerance* options.

`-ignore` *`ign_list`*

Specifies a list of categories of slack or arc data to be excluded from the comparison. The following list describes the valid values:

max - Excludes all max_rise and max_fall delay type arcs from the slack and transition_time sections. The capacitance section is not examined because it contains only maximum information.

min - Excludes all min_rise and min_fall delay type arcs from the slack and transition_time sections.

unconstrained - Excludes all paths that are unconstrained for both reference and compare data files.

pass - Excludes all passing comparisons, so only fails are displayed.

async_default - Excludes the comparison of paths in the async_default path group. This option is obsolete and is replaced by specifying the *recover* and *removal* values of the *-include* option.

clock_gating - Excludes comparison of paths with the clock_gating_default path group. This option is obsolete and is replaced by specifying the *clock_gating_setup* and *clock_gating_hold* of the *-include* option.

input_to_register - Excludes comparison of input-to-register paths. This option is obsolete and is replaced by specifying the *setup* and *hold* values of the *-include* option.

register_to_output - Excludes comparison of register-to-output path. This option is obsolete and is replaced by specifying the *max_seq_delay* and *min_seq_delay* of the *-include* option.

input_to_output - Excludes comparison of combinational paths. This option is obsolete and is replaced by specifying the *max_combo_delay* and *min_combo_delay* of the *-include* option.

`-include` *`cmp_list`*

Specifies a list of data sections from the *write_interface_timing* command output to be included in the comparison; only those sections are compared. By default, all sections are compared. The following list describes the keywords that are accepted in the *cmp_list* as valid:

slack - Specifies that only worst-case slacks are to be compared.

arc - Specifies that only arc values are to be compared.

transition_time - Specifies that only actual transition times on ports are to be compared.

capacitance - Specifies that only actual capacitances on ports are to be compared.

design_rules - Specifies that only design rules on ports are to be compared.

missing_arcs - Specifies that only arcs in the reference timing file that are missing from the compare timing file are to be included in the report. If it is used together with other sections, the report includes the intersection of *missing_arcs* and the other specified sections, not the union of them.

max_combo_delay - Specifies that only maximum combination delay arc type paths are to be reported.

min_combo_delay - Specifies that only minimum combination delay arc type paths are to be reported.

max_seq_delay - Specifies that only maximum sequential delay arc type paths are to be reported.

min_seq_delay - Specifies that only minimum sequential delay arc type paths are to be reported.

setup - Specifies that only setup arc type paths are to be reported.

hold - Specifies that only hold arc type paths are to be reported.

recovery - Specifies that only recovery arc type paths are to be reported.

removal - Specifies that only removal arc type paths are to be reported.

clock_gating_setup - Specifies that only clock gating setup arc type paths are to be reported.

clock_gating_hold - Specifies that only clock gating hold arc type paths are to be reported.

`-quiet`

Specifies an obsolete option. This option is now obsolete and has been replaced by the *-session quiet* option.

`-nosplit`

Prevents line-splitting. This is most useful for performing difference comparisons on previous scripts or for post processing the script.

`-sort_by_worst`

Specifies that the output is to be sorted from negative to positive, with the worst failure first. After the FAIL section, all PASS lines are sorted alphabetically within each path attribute. By default, the output follows the order of the reference write_interface_timing data, which is sorted alphabetically within each path attribute (c paths, i paths including a and g, then o paths).

`-session` *`session_name`*

Controls the information that is printed to the session transcript. Note that this option affects only the session transcript, not the information written to the output file. The following list describes the keywords that are accepted for *session_name*:

quiet - Prevents information, warning, and error messages from appearing in the session transcript. Use this option when you want to see only the success/failure return code without any MV messages. The *-session* option used with this keyword replaces the now-obsolete *-quiet* option.

summary - Prints only the summary pass/fail count to the session transcript.

full - Prints the detailed report either to the session transcript or to a file if you specify the *-output* option. This is the default behavior.

`-significant_digits` *`digits`*

Specifies the number of digits to the right of the decimal point that are to be reported following the method used in the *report_timing* command. Allowed

values are 0-13. The default is 2. Use this option if you want to override the default. See below for a more detailed description of significant digits.

Description

This command compares two interface timing files (called the "reference" and "comparison" files) previously generated by the *write_interface timing* command. The result is PASS if the timing parameter values in the two files are the same or within a specified tolerance. The result is FAIL if both columns have data and the tolerance is exceeded. If all comparisons in the report are PASS, the return code for the *compare_interface_timing* command is 0. If one or more comparisons result in FAIL, the return code is 1. Any other return code (or no return code) indicates a command error. If either file format does not conform to the *write_interface_timing* standard, the command issues a warning message.

The *compare_interface_timing* command lets you specify the input and output files for the comparison, the types of paths and timing parameters to compare or not compare, and the allowed tolerance levels that trigger comparison failures.

If the reference and comparison files were generated using different contexts then the two files might contain different timing arcs along with different slacks. If an arc exists in the comparison file but not in the reference file, the *compare_interface_timing* command ignores the extra arcs. If an arc exists in the comparison file but not in the reference file, the *compare_interface_timing* command reports the slack value for the reference file, but either "N.C." or "----" for the comparison file slack. The slack difference is marked as "----", and the status is FAIL. The comparison slack is marked as "----" if the arc is not clock gating or recovery or removal.

The slack for a missing arc is mark as "N.C.", meaning not critical, if it exists in the reference but not in the comparison file, and the arc type is clock gating or recovery or removal. These arcs fall into the **clock_gating_default** or **asynchronous_default** clock groups, respectively. Each of these groups can have several clocks in them, and under certain circumstances the *write_interface_timing* command sees the path to one clock as critical for the reference view and report that arc and slack, but another clock as critical in the comparison view and report that arc and slack.

The *compare_interface_timing* command deals with significant digits in two distinct ways. First, the internal computations are limited to the minimum precision of the two input files. This means that if the *ref_timing_file* was generated with three digits and the *cmp_timing_file* with four, the *compare_interface_timing* command uses three digits for all internal computations.

It is important to make the tolerances larger than a unit of the computational significant digits, or the *compare_interface_timing* command could generate false FAIL and PASS messages. This can happen because of rounding. For example, if the internal significant digits is two, and the absolute tolerance is set to 0.01, a difference of 0.014 is rounded to 0.01 and reported as a PASS even though it is outside the tolerance. Similarly, if the

absolute tolerance is set to 0.009, a difference of 0.009 is rounded to 0.01 and reported as FAIL even though it is within tolerance. You should ensure that the number of digits of accuracy of the input files is greater than the accuracy of the tolerances.

A second use of significant digits is in report generation, which is limited to the minimum of the input precision and the significant digits that you specified. You specify the significant digits for the report through the *-significant_digits* option. For example, if the value for the *-significant_digits* option is five but the *ref_timing_file* was generated with four digits, the report is limited to four digits. Note that this use of significant digits does not affect the PASS or FAIL status, but a reported difference could look like it should have a different status if the difference is close to a tolerances and the difference rounded when writing the report.

Each tolerance is a list of either one or two floating point numbers. All tolerances are inclusive, and the sign of the values has no effect. If there is only one number, it serves to specify both the plus and minus tolerances. The plus tolerance is the absolute value of the number, and the minus tolerance is the inverse of the plus. For example, *atol_list* equal to *0.1* indicates that time differences of less than 0.1 and greater than -0.1 are considered to be passing.

If there are two numbers, the first specifies the minus bound and the second the plus. The minus tolerance is the inverse of the absolute value of the first number, and the plus tolerance is the absolute value of the second number. Having a pair of numbers allows different tolerances to be applied for negative and positive errors, which can be interpreted as optimism and pessimism depending on the arc type. For example, an *atol_list* equal to *{0.2 -0.3}* indicates that data differences of less than 0.3 and greater than -0.2 are considered passing.

Examples

The following example shows a variety of comparisons. Each item in parentheses (for example, L1, L2) represents the latch level associated with the slack number directly to the left. These values are included in the report wherever there is an *L#* attribute in the *write_interface_timing* output. At the end of the report is a summary of the total comparisons passed and failed broken down by data section.

```
pt_shell> compare_interface_timing demo_net.rpt demo_etm.rpt \\
            -include arc -percent_tolerance {10 5}

****************************************
Command: compare_interface_timing
         demo_net.rpt demo_etm.rpt
         -include arc
         -percent_tolerance 10.0 5.0
Design : top
 ...
****************************************

                Arc                          Arc Value
```

```
From     To       Type                      Ref              Cmp     %Error
 Status
-------------------------------------------------------------------------
----
in(r)    out(r)   max_combo_delay          2.63             2.21     15.97
 FAIL
in(f)    out(r)   max_combo_delay          2.28             2.32     -1.75
 PASS
in(r)    out(f)   min_combo_delay          0.17             0.30    -76.47
 FAIL
in(f)    out(f)   min_combo_delay          0.51             0.51      0.00
 PASS
in(r)    clk(r)   setup                    2.63     (L1)    2.63      0.00
 PASS
in(f)    clk(r)   setup                    2.29     (L2)    2.29      0.00
 PASS
in(r)    clk(f)   hold                     0.17             0.17      0.00
 PASS
clk(r)   out(r)   max_seq_delay           17.79            17.79      0.00
 PASS
clk(r)   out(f)   max_seq_delay           18.20            18.20      0.00
 PASS
clk(f)   out(r)   min_seq_delay            2.21             2.21      0.00
 PASS
clk(f)   out(f)   min_seq_delay            1.80             1.80      0.00
 PASS


         Totals  Arc Value Transition Time   Capacitance  Rules
-----------------------------------------------------------------
Passed   9       9         0                 0            -
Failed   2       2         0                 0            0
Total    12      12        0                 0            0
```

See Also

- write_interface_timing

---

compile_scaling_library_group_data

Generates the scaling.db file for the library passed in.

Syntax

string *compile_scaling_library_group_data*

```
library_name
[-config_file DBFile_name]
```

Feedback

Data Types

library_name	list
config_file	string

Arguments

library_name

Collection of libraries in memory for which the companion.db files are to be created.

-config_file *DBFile_name*

Specifies writing data to DBFile.

Description

The *compile_scaling_library_group_data* command pre-computes all derived library data that is needed during *update_timing* and deposits it a "scaling.db" file. This file is consumed in *update_timing* if it can be found in the *search-path* and allows to use the precomputed derived libray data instead of computing it on the fly.

The scaling.db file is generated in the current directory.

Examples

The following snippet would typically be used during library data prep to generate the scaling.db file

```
pt_shell> compile_scaling_library_group_data 1.db -config_file 1_fusion
```

---

complete_net_parasitics

Completes partial parasitics annotated on all nets of the current design.

Syntax

string *complete_net_parasitics*

[-complete_with *completion_type*]

Data Types

completion_type	string

Arguments

-complete_with *completion_type*

Indicates that a net with partially annotated parasitics and missing segments is to be completed by inserting capacitances and resistances according to the *completion_type* value. Allowed values are *zero* (the default), which completes

the net by inserting zero capacitances and resistances, and *wlm*, which completes the net by inserting capacitances and resistances derived from wire load models.

This option is equivalent to the *read_parasitics -complete_with* command.

Description

This command completes all nets in the design that have incomplete RC networks annotated from SPEF or SPF files.

If lumped parasitics (set using the *set_load* or *set_resistance* command) already exist on any of the nets, they are not overwritten; however, a warning message is issued.

In a hierarchical flow where you read in multiple parasitics files, do not use this command until all parasitics files are read in, or you will create incorrect data for temporarily incomplete networks at block boundaries.

Note: The *complete_net_parasitics* and *read_parasitics -complete_with* commands complete a net only if all missing segments are between two pins, and only if the nets are partially annotated. Nets are not affected if they are fully annotated or have no annotation at all. Also, the net must be hierarchical, so that if the parasitics for the block-level parts of a net are missing, those parasitics could exist in the top-level net. If any of these conditions are not met, you must manually correct the SPEF or DSPF file.

Use the *report_annotated_parasitics* command to view how the parasitics are completed.

After the completion of the nets, there is no direct way to remove only the completed segments. You can remove all parasitics read and annotated with the *read_parasitics* and *complete_net_parasitics* commands using the *remove_annotated_parasitics* command. You then reread the previously annotated parasitics using the *read_parasitics* command. The *reset_design* command removes all attributes from the current design, including annotated parasitics.

Examples

The following example completes the partially annotated parasitics from the file speffile.spef, by inserting zero capacitances and resistances.

```
pt_shell> read_parasitics speffile.spef
pt_shell> complete_net_parasitics -complete_with zero
```

The following example completes any partially annotated parasitics after reading in hierarchical parasitics.

```
pt_shell> read_parasitics blkA.spf -path blkA
pt_shell> read_parasitics blkB.spf -path blkB
pt_shell> read_parasitics top_level.spf
pt_shell> complete_net_parasitics -complete_with zero
```

Feedback

See Also

- read_parasitics
- remove_annotated_parasitics
- report_annotated_parasitics
- reset_design

---

compute_metrics

Performs RTL Power analysis.

Syntax

string *compute_metrics*

```
[-power]
[-elaborated_block eblock]
[-reuse workspace]
```

Data Types

```
eblock      block_name
workspace      directory
```

Arguments

`-power`

This option performs FAST compile in a separate RTL Architect process (under the hood) and then calculates RTL power metrics for the compiled design.

`-elaborated_block eblock`

This option compiles the elaborated block *eblock* and does RTL power analysis for the block.

`-reuse workspace`

This option loads a previous existing *workspace* directory and automatically performs RTL power analysis.

Description

This is a high level command that performs power analysis starting from rtl.

Feedback

See Also

- update_power
- update_metrics
- report_rtl_metrics

---

configure_sms_scenarios

Configures SMS scenarios for selective SMVA/SMS analysis.

Syntax

string *configure_sms_scenarios*

```
[-disable sms_scenarios_list]
[-enable sms_scenarios_list]
```

Data Types

```
sms_scenarios_list       collection
```

Arguments

`-disable sms_scenarios_list`

Specifies a collection of SMS scenarios to disable. The specified SMS scenarios will be ignored during timing analysis and reporting. This collection is created by *get_sms_scenarios*.

`-enable sms_scenarios_list`

Specifies a collection of SMS scenarios to re-activate. The specified SMS scenarios will be re-activated for a subsequent full timing analysis and reporting. Only previously disabled SMS scenarios can be specified for re-activation. This collection is created by *get_sms_scenarios*.

Description

The *configure_sms_scenarios* command configures SMS scenarios for selective SMVA/SMC analysis. PrimeTime analyzes by default all relevant and active SMS scenarios. Specific scenarios can be disabled using the -disable command option, thus reducing the SMS scenarios analysis space. Disabled scenarios are not considered for timing analysis and subsequent reporting. Disabled scenarios can be re-activated for analysis using the -enable command option. This command is only available with SMVA/SMC analysis.

Examples

The following example specifies 3 voltage reference names {high,low,typ} for each one of the supply groups SN1 and SN2. PrimeTime automatically considers all the

relevant combinations of voltage levels, in this case up to nine prior the use of the configure_sms_scenarios command. The example disables all SMS scenarios for which the supply group SN1 is at 'typ' voltage and SN2 is at 'typ' voltage.

```
pt_shell> set_voltage_levels SN1 -reference_names { high low typ }
pt_shell> set_voltage -o SN1 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN1 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN1 -reference_name typ 0.9 -min 1.0
pt_shell> set_voltage_levels SN2 -reference_names { high low typ }
pt_shell> set_voltage -o SN2 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN2 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN2 -reference_name typ 0.9 -min 1.0

pt_shell> report_sms_scenarios -disabled
****************************************
Report : report_sms_scenarios
 -disabled
Design : top
Version: Q-2019.12-DEV-20191027
Date   : Mon Oct 28 17:13:07 2019
****************************************

All scenarios are currently non-ignored.

pt_shell> configure_sms_scenarios -disable [get_sms_scenario
 -supply_group SN1 -reference_name typ]
pt_shell> configure_sms_scenarios -disable [get_sms_scenario
 -supply_group SN2 -reference_name typ]

pt_shell> report_sms_scenarios -disabled
****************************************
Report : report_sms_scenarios
 -disabled
Design : top
Version: Q-2019.12-DEV-20191027
Date   : Mon Oct 28 17:13:07 2019
****************************************

  Current Ignored Scenarios:
------------------------------------------------------------
 SN2:high low, SN1:typ
 SN2:typ
```

The following example continues from the previous example and re-activates the SMS scenario having SN1 at 'typ' voltage and SN2 at 'low' voltage.

```
pt_shell> configure_sms_scenarios -enable [get_sms_scenarios
 -supply_group SN1 -reference_names typ -supply_group SN2
 -reference_names low]

pt_shell> report_sms_scenarios -disabled
```

Feedback

```
****************************************
Report : report_sms_scenarios
 -disabled
Design : top
Version: Q-2019.12-DEV-20191027
Date   : Mon Oct 28 17:28:06 2019
****************************************

  Current Ignored Scenarios:
------------------------------------------------------------
 SN2:high, SN1:typ
 SN2:typ
```

See Also

- get_sms_scenarios
- report_sms_scenarios
- set_voltage_levels
- set_voltage

---

connect_net

Connects a net to specified pins or ports.

Syntax

int *connect_net*

net
object_spec

Data Types

net	string
object_spec	list

Arguments

net

Specifies the name of the net to which the pins and ports are to be connected.

object_spec

Specifies a list of pins or ports to connect to *net*.

Description

Connects pins and ports to a *net* in the current design. Like all other netlist editing commands, for the *connect_net* command to succeed, all of its arguments must succeed.

If any arguments fail, the netlist remains unchanged. The *connect_net* command returns a 1 if successful and a 0 if unsuccessful.

The *net* and each pin and port specified in the *object_spec* must be in scope; that is, at or below the current instance.

There are two rules for the *connect_net* command:

- You cannot connect *net* to a pin that is already connected.
- You cannot connect across a hierarchical boundary. For example, you cannot connect a port to a net at a hierarchical level other than the top of the design.

Examples

The following example creates a cell and a net and then connects the new net to the output of the new cell:

```
pt_shell> create_net new_net1
Information: Created net 'new_net1' in design 'top'. (NED-016)
1

pt_shell> create_cell u1 class/AN2
Information: Created cell 'u1' in design 'top'. (NED-014)
1

pt_shell> connect_net new_net1 u1/Z
Information: Connected 'u1/Z' to 'new_net1'. (NED-018)
1
```

See Also

- create_cell
- create_net
- disconnect_net
- size_cell
- swap_cell
- write_changes

---

connect_power_domain

Connects power net information for the specified power domains.

Feedback

Syntax

status *connect_power_domain*

```
power_domains
[-primary_power_net power_net]
[-primary_ground_net ground_net]
[-backup_power_net power_net]
[-backup_ground_net ground_net]
[-internal_power_net internal_power_net]
[-internal_ground_net internal_ground_net]
```

Data Types

power_domains	collection
power_net	string
ground_net	string
internal_power_net	string
internal_ground_net	string

Arguments

power_domains

Specifies the power domains for which to make the power net information connections.

-primary_power_net *power_net*

Specifies the primary power net information for the power domains.

-primary_ground_net *ground_net*

Specifies the primary ground net information for the power domains.

-backup_power_net *power_net*

Specifies the backup power net information for the power domains.

-backup_ground_net *ground_net*

Specifies the backup ground net information for the power domains.

-internal_power_net *internal_power_net*

Specifies the internal power net information for the power domains.

-internal_ground_net *internal_ground_net*

Specifies the internal ground net information for the power domains.

Description

The *connect_power_domain* command creates logical power connections for the specified power domains. All cells in the power domain inherit the power connections. You can

Feedback

override the inherited power connections for a cell by using the *connect_power_net_info* command.

The primary power and ground nets are used to define the main power connections for the power domain.

The backup power and ground nets are used to define power connections for always-on logic, retention registers, isolation cells, and enable-level shifter cells.

The internal power and ground nets are used to connect power nets to the switching cells present inside the power domain.

Examples

The following example connects power net information to the TOP_DOMAIN and SUB_DOMAIN power domains.

```
prompt> connect_power_domain TOP_DOMAIN \\
   -primary_power_net T_VDD \\
   -primary_ground_net T_VSS
1
prompt> connect_power_domain SUB_DOMAIN \\
   -primary_power_net A_VDD  \\
   -primary_ground_net A_VSS \\
   -backup_power_net VDD_Backup \\
   -backup_ground_net VSS_Backup
1
```

See Also

- connect_power_net_info
- create_power_domain
- get_power_domains
- report_power_domain

---

connect_power_net_info

Connects the specified power net information to the specified power pins.

Syntax

status *connect_power_net_info*

```
object_list
-power_pin_name power_pin_name
-power_net_name power_net_name
```

Feedback

Data Types

```
object_list         list
power_pin_name      string
power_net_name      string
```

Arguments

object_list

Specifies the leaf cells for which the power net information connections are made.

-power_pin_name *power_pin_name*

Specifies the name of the power pin.

-power_net_name *power_net_name*

Specifies the name of the power net.

Description

The *connect_power_net_info* command makes power net information connections for a specific power pin of a leaf cell. The pin-level connection overrides the domain-level connections made with the *connect_power_domain* command.

See the *report_power_pin_info* man page for more information.

Examples

The following example connects the power net information for the PWR pin of the PD0_INST/I0 cell.

```
prompt> connect_power_net_info PD0_INST/I0 \\
   -power_pin_name PWR -power_net_name exp_VDD
1
```

See Also

- connect_power_domain
- report_power_pin_info

---

connect_supply_net

Connects the supply net to specified supply ports or pins. This command is part of UPF definition of virtual power and ground network.

Feedback

Syntax

```
status connect_supply_net
  -ports list
  supply_net_name
```

Data Types

```
list                list
supply_net_name     string
```

Arguments

`-ports list`

Specifies supply ports to be connected to the supply net. The name is hierarchical name.

`supply_net_name`

Specifies the name of the supply_net to be connected. The name should be a simple (nonhierarchical) name. The net must exist in the current scope. This argument is required. If you use this option in UPF mode, it specifies which supply ports or instance power ground pins are to be connected with the supply net.

Description

This command provides an explicit connect of a supply net to any supply ports or pins and overrides (has higher precedence than) the auto-connection semantics that might otherwise apply. If a design element is not connected explicitly to any supply net using the *connect_supply_net* command, it shares the primary power/ground supply net with the power domain it belongs to.

The instance that contains the pin must also be in the extend of the same power domain. A supply port cannot connect to two nets in the same domain. Note that the commands does not specify whether the net connects to the inside or outside side of the port. The port side is auto-derived from the net domain and port scope.

Note that explicit connections cannot be made to cell PG pins that are created based on default PG pin names by the tool.

Examples

The following example connects a VSS supply net with VSS supply port and ground pin of cell INST_1/u1:

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net VSS -domain PD1
VSS
prompt> create_supply_port VSS -domain PD1
```

```
VSS
prompt> connect_supply_net VSS -ports VSS
1
prompt> connect_supply_net VSS -ports INST_1/u1/GND
1
```

See Also

- create_power_domain
- create_supply_net
- create_supply_port
- get_supply_nets
- get_supply_ports
- report_supply_net

---

convert_spef_to_gpd

Converts a SPEF file into the GPD (Galaxy Parasitics Database) binary format.

Syntax

status *convert_spef_to_gpd*

```
spef_file_name
  gpd_dir
```

Data Types

```
spef_file_name      string
gpd_dir             string
```

Arguments

spef_file_name

Specifies the SPEF file to be converted to GPD. Input SPEF file can be compressed/uncompressed.

gpd_dir

Specifies the directory where the GPD data files are written.

Description

This command converts a SPEF file into the GPD (Galaxy Parasitic Database) format, which is written as a set of files in the output directory.

Feedback

Examples

The following example converts a compressed SPEF file xx.spef.gz and generates the output GPD in the mygpd directory.

```
pt_shell> convert_spef_to_gpd xx.spef.gz mygpd
```

See Also

- read_parasitics

---

copy_collection

Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged.

Syntax

collection *copy_collection*

```
        collection1
```

Data Types

```
collection1          collection
```

Arguments

collection1

Specifies the collection to be copied. If an empty string is used for the *collection1* argument, the command returns the empty string (a copy of the empty collection is an empty collection).

Description

This command is no longer needed and is provided only to make old scripts work without modification.

Examples

The following example from PrimeTime shows the result of copying a collection. Functionally, it is not much different that having multiple references to the same collection, except it is slower.

```
pt_shell> set c1 [get_cells "U1*"]
{"U1", "U10", "U11", "U12"}
pt_shell> set c2 [copy_collection $c1]
{"U1", "U10", "U11", "U12"}
```

```
pt_shell> unset c1
pt_shell> query_objects $c2
{"U1", "U10", "U11", "U12"}
```

See Also

- collections

---

cputime

Retrieves the overall user time associated with the current pt_shell process.

Syntax

float *cputime*

[*format*]

Data Types

format string

Arguments

format

This argument takes a *print* style formatting string for a floating point number.

Description

This command returns the accumulated user time, in seconds, for the current pt_shell process. If the *format* string is omitted, an integer number of seconds is returned. If the *format* string is specified, a floating point number is returned. The number includes fractions of a second elapsed and is formatted in accordance with given specifier. For detailed information about formatting a floating point argument, see the *print* UNIX man page.

Examples

```
pt_shell> cputime "%g"
3.74
```

See Also

- mem

---

create_cell

Creates cells in the current design.

Feedback

Syntax

```
status create_cell
  [-libraries lib_spec]
  [-exact]
  cell_list
  lib_cell
```

Data Types

```
lib_spec          string
cell_list         list
lib_cell          string
```

Arguments

`-libraries lib_spec`

If this option is specified, PrimeTime resolves the *lib_cell* option from the libraries contained in only the *lib_spec* option. Libraries are searched in the order in which they appear in *lib_spec*. The *lib_spec* value can be a list of library names or collections of libraries loaded into PrimeTime. You can obtain the latter by using the *get_libs* command. You cannot specify this option if a full library cell name has been specified.

`-exact`

Indicates that names are to be used exactly as specified. Use this option if you want to create a cell that contains a hierarchy or wildcard character as part of the cell name. For more information, see the Naming Cells With Special Characters section.

cell_list

Specifies a list of cells to be created.

lib_cell

Specifies the name of the library cell to which the specified cells are to be linked. The *lib_cell* option can be a library cell object or the name of a library cell. You can obtain the former by using the *get_lib_cells* command. The latter can either be the full library cell name, such as *lib_name/lcell_name*, or the just the base name of the library cell, such as *lcell_name*. You cannot specify the *-libraries* option and explicitly specify the full library cell name. If you invoke PrimeTime with the *-multi_scenario* option, only the library cell base name must be used. For more information, see the Resolved Library Cells section.

Description

This command creates new cells in the current design. Like all other netlist editing commands, for the *create_cell* command to succeed, all of its arguments must succeed. If

any arguments fail, the netlist remains unchanged. The *create_cell* command returns a 1 if successful and a 0 if unsuccessful.

Cells are created in scope; that is, at or below the current instance. Cell names are specified as for other commands, using the full hierarchical name relative to the current instance, as in the following example:

```
create_cell u1/u2/u3 class/AN2
```

This command attempts to create a cell named u3 in the hierarchical block u1/u2. The name to the right of the last hierarchical separator is the actual cell name, and the remainder is sent to a *search engine* to find a hierarchical block in which to create the new cell. The operation fails if any of the following are true:

- The search engine cannot find u1/u2.
- The search engine finds multiple blocks that match u1/u2.
- The search engine finds a leaf cell matching u1/u2.

Currently, you cannot create new hierarchy; that is, you cannot create a cell that is an instance of a design. The *lib_cell* must be a leaf library cell.

After creating a cell, you can connect nets to its pins with the *connect_net* command. You can remove cells with the *remove_cell* command.

Resolving Library Cells

If the *lib_cell* has been specified in base name only format, such as without a library from which to resolve it from, PrimeTime resolves the library cell according to the following methodology.

If the *-libraries* option is specified, PrimeTime searches for library cells only in the libraries contained within the *lib_spec*.

Alternatively, if the *-libraries* option is not specified, PrimeTime searches for the library cell in the libraries contained within the *link_path* variable.

The first library cell that is found is used.

Naming Cells With Special Characters

If you want to create a cell whose name contains the current hierarchical separator or wildcard characters used by the search engine, you must do the following:

- Use the *current_instance* command to change the scope to the block in which you want the new cell created.
- Use the *create_cell -exact* command to create the cell.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Feedback

Examples

In the following example, cells are created in the current instance, and at a level below the current instance. Currently, u1 is an instance of a design that has multiple instances. Therefore, it is uniquified as part of the editing operation.

```
pt_shell> create_cell t1 class/AN2
Information: Created cell 't1' in design 'top'. (NED-014)
1

pt_shell> create_cell u1/t1 class/AN2
Uniquifying 'u1' (block1) as 'block1_0'.
Information: Created cell 't1' in 'top/u1'. (NED-014)
1
```

The following example creates cell a/b in the hierarchical block u1/u2. The first attempt fails, because the *exact* option was omitted. Here, u1/u2 is an instance of a design that has multiple instances; therefore, it is uniquified as part of the editing operation. The u1 was already uniquified in the previous example.

```
pt_shell> current_instance u1/u2
u1/u2

pt_shell> create_cell  a/b class/AN2
Error: Could not create cell 'a/b':
        a not found. (NED-041)
Error: No changes made. (NED-040)
0

pt_shell> create_cell a/b class/AN2  -exact
Uniquifying 'u1/u2' (block2) as 'block2_0'.
Information: Created cell 'a/b' in 'top/u1/u2'. (NED-014)
1
```

In the following example, multiple cells are created using library cells chosen from user-specified libraries using only the library cell base name. The 'lib1' library is searched first as it appears in the *lib_spec* list first and then 'lib2' is searched.

```
pt_shell> create_cell -libraries {lib1 lib2} {u1 u2 u3} ND2
Information: Created cell 'u1' in design 'middle' with 'lib1/ND2'.
 (NED-014)
Information: Created cell 'u2' in design 'middle' with 'lib1/ND2'.
 (NED-014)
Information: Created cell 'u3' in design 'middle' with 'lib1/ND2'.
 (NED-014)
1
```

Feedback

See Also

- connect_net
- create_net
- current_instance
- remove_cell
- size_cell
- swap_cell
- write_changes
- link_path

---

create_clock

Creates a clock object.

Syntax

string *create_clock*

```
-period period_value
[-name clock_name]
[-waveform edge_list]
[-add]
[-sms_scenarios sms_scenarios_list]
[-comment comment_string]
[source_objects]
```

Data Types

period_value	float
clock_name	string
edge_list	list
source_objects	list
sms_scenarios_list	collection
comment_string	string

Arguments

`-period period_value`

Specifies the clock period in library time units. This is the minimum time over which the clock waveform repeats. The *period_value* type must be greater than or equal to zero.

`-name` *clock_name*

Specifies the name of the clock being created. If you do not use this option, the clock gets the same name as the first clock source specified in the *source_objects* list. If you did not specify the *source_objects* list, you must use the *-name* option, which creates a virtual clock not associated with a port or pin. You can use both the *-name* and *source_objects* options to give the clock a more descriptive name than the first source pin or port. If you specify the *-add* option, you must use the *-name* option and the clocks with the same source must have different names.

`-waveform` *edge_list*

Specifies the rise and fall edge times of the clock waveforms of the clock, in library time units, over an entire clock period.

The first time in the *edge_list* is a rising transition, typically the first rising transition after time zero. There must be an even number of edges, and they are assumed to be alternating rise and fall.

The edges must be monotonically increasing, except for a special case with two edges, where fall can be less than rise. The numbers should represent one full clock period.

If you do not specify this option, a default waveform is assumed, which has a rise edge of 0.0 and a fall edge of *period_value*/2.

`-add`

Specifies whether to add this clock definition to the existing clock or to overwrite it. Use this option to capture the case where multiple clocks must be specified on the same source for simultaneous analysis with different clock waveforms. When you specify this option, you must also use the *-name* option.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because PrimeTime must explore all possible combinations of launch and capture clocks. Use the *set_clock_groups* or *set_false_path* command to disable unwanted clock combinations.

`-comment` *comment_string*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC, such as to track the methodology or tool that originated the command.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios for which this clock specification is analyzed. When this option is not given, the clock specification applies for all SMS scenarios. PrimeTime requires that clock waveforms to be specified

for all applicable SMS scenarios. It is thus recommended to first specify the clock waveforms without this option, followed by the desired waveforms at specific SMS scenarios using this option. This collection is created by *get_sms_scenarios*.

`source_objects`

Specifies the objects used as sources of the clock. The sources can be ports, pins, or nets in the design. If you do not use this option, you must use the *-name* option, which creates a virtual clock not associated with a port, pin, or net. If you specify a clock on a pin that already has a clock, the new clock replaces the old one unless you use the *-add* option. When a net is used as the source, the first driver pin of the net is the actual source used in creating the clock.

Description

Creates a clock object. It is created in the current design and is applied to the specified *source_objects* list. If you do not specify a *source_objects* list, but give a *clock_name* value, a virtual clock is created. You can create a virtual clock to represent an off-chip clock for input or output delay specification. For more information about input and output delay, see the *set_input_delay* and *set_output_delay* man pages.

If one of the *source_objects* values is already the source of a clock, the source is removed from that clock. This clock is eliminated if it has just one source.

The *create_clock* command also defines the waveform for the clock. The clock can have multiple pulses per period.

The *create_clock* command is used together with the *set_clock_latency*, *set_clock_uncertainty*, *set_propagated_clock*, and *set_clock_transition* commands to specify properties of clock networks. By default, a new path group is created for the clock. This new path group brings together the endpoints related to this clock for cost function calculation. To remove the clock from its assigned group, use the *group_path* command to reassign the clock to another group or to the default path group. For more information, see the *group_path* man page.

The new clock has ideal clock latency and transition time; no propagated delay through the clock network is assumed and a transition time of zero is used at the clock source pin. To enable propagated latency for a clock network, use the *set_propagated_clock* command. To set an estimated latency, use the *set_clock_latency* command.

To show information about clocks in the design, use the *report_clock* command. To create a collection of clocks matching a pattern and optionally matching filter criteria, use the *get_clocks* command.

To undo *create_clock*, use the *remove_clock* command.

Feedback

Examples

The following example creates a clock on a port named *PHI1* with a period of *10.0*, a rise at *5.0*, and a fall at *9.5*.

```
pt_shell> create_clock PHI1 -period 10 -waveform { 5.0 9.5 }
```

The following example shows a clock named *PHI2* with a falling edge at *5* and a rising edge at *10* with a period of *10*. Because the edges for the *-waveform* option should be ordered as first rise, then fall, and should increase in value, the fall edge can be given as *15*. This places the next falling edge after the first rise edge at *10*.

```
pt_shell> create_clock PHI2 -period 10 -waveform { 10 15 }
```

The following example creates a virtual clock *PHI2* with a period of *10.0*, a rise at *0.0*, and a fall at *5.0*.

```
pt_shell> create_clock -name PHI2 -period 10 -waveform {0.0 5.0}
```

The following example creates a clock named *clk2* with multiple sources.

```
pt_shell> create_clock -name clk2 -period 10 \\
-waveform {2.0 4.0} {clkgen1/Z clkgen2/Z clkgen3/Z}
```

The following example creates a clock named *CLK* on pin *u13/Z* with a period of *25*, a fall at *0.0*, a rise at *5.0*, a fall at *10.0*, a rise at *15.0*, and so on.

```
pt_shell> create_clock u13/Z -name CLK -period 25 -waveform { 5 10 15 25}
```

See Also

- all_clocks
- get_clocks
- group_path
- remove_clock
- report_clock
- set_clock_latency
- set_clock_uncertainty
- set_input_delay
- set_output_delay
- set_propagated_clock
- set_clock_transition

Feedback

create_command_group

Creates a new command group.

Syntax

string *create_command_group* [-info *info_text*] *group_name*

Arguments

`-info` *info_text*

Help string for the group

group_name

Specifies the name of the new group.

Description

The *create_command_group* command is used to create a new command group, which you can use to separate related user-defined procedures into functional units for the online help facility. When a procedure is created, it is placed in the "Procedures" command group. With the *define_proc_attributes* command, you can move the procedure into the group you created.

The *group_name* can contain any characters, including spaces, as long as it is appropriately quoted. If *group_name* already exists, *create_command_group* quietly ignores the command. The result of *create_command_group* is always an empty string.

Examples

The following example demonstrates the use of the *create_command_group* command:

```
prompt> create_command_group {My Procedures} -info "Useful utilities"

prompt> proc plus {a b} { return [expr $a + $b] }

prompt> define_proc_attributes plus -command_group "My Procedures"

prompt> help
My Procedures:
  plus

 ...
```

See Also

- define_proc_attributes
- help

Feedback

create_drc_error

Creates a physical DRC error object.

Syntax

```
collection create_drc_error
-error_data drc_error_data
-error_type drc_error_type
[-status error | fixed | ignored | waived]
[-polygons shape_list]
[-polylines shape_list]
[-endpoints coord_list]
[-points shape_list]
[-objects objects]
[-must_fix]
[-direction direction]
[-required_spacing distance]
[-actual_spacing distance]
[-width_required distance]
[-height_required distance]
[-shape_uses shape_uses]
[-pin_edge pin_edge]
[-information information]
```

Data Types

```
drc_error_data  collection
drc_error_type  collection
shape_list      list
coord_list      list
objects         collection
direction       string
distance        float
shape_uses      list
pin_edge        integer
information     string
```

Arguments

`-error_data drc_error_data`

Specifies the error data in which to create the physical DRC error.

`-error_type drc_error_type`

Specifies the error type for which to create the physical DRC error. The error type value becomes available as the *error_type* attribute for the error.

`-status error | fixed | ignored | waived`

Sets the initial status of the new error. Available values are: *error, fixed, ignored,* and *waived*. If omitted, the status value *error* is used. The status value becomes available as the *status* attribute for the error.

`-polygons` *`shape_list`*

Polygon or rectangle shapes which describe the DRC error. The *shape_list* argument is a list of shapes where each shape is a list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the *polygons* attribute for the error.

`-polylines` *`shape_list`*

Describes polylines shapes for DRC error. The *shape_list* argument is list of shapes where each shapes consist of list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the *polylines* attribute for the error.

`-endpoints` *`coord_list`*

Specifies the two endpoints of distance shape of violation such as spacing or open error. The *coord_list* argument is a list of two x- and y-coordinate pairs: { {*x y*} {*x y*} }. If more than two points are given, only the first two are used and the rest are ignored. This option is a required option when create open error. The coordinates become available as the *endpoints* attribute for the error.

`-points` *`shape_list`*

Specifies points as shape of violation when necessary. The *shape_list* argument is a list of x- and y-coordinate pairs: { {*x y*} {*x y*} ... }. The coordinates become available as the *points* attribute for the error.

`-objects` *`objects`*

Specifies design objects associated with the new error. The objects become available as the *objects* attribute for the error.

`-must_fix`

Sets the *must fix* flag on the error. The default is false. The must fix value becomes available as the *must_fix* attribute for the error.

`-direction` *`direction`*

Sets the direction of a spacing violation. The option argument is ignored if the associated error type is not in the spacing class. The error class for the error type is set when it is created. The direction is used in rendering the spacing violation in the GUI layout view. The direction value becomes available as the *direction* attribute for the error.

`-required_spacing` *`distance`*

Sets the required spacing distance which was not met in a spacing violation. The option argument is ignored if the associated error type is not in the spacing class. The error class for the error type is set when it is created. The required distance is used to provide a more detailed rendering of the spacing violation in the GUI layout view. The required spacing value becomes available as the *required_spacing* attribute for the error.

`-actual_spacing` *`distance`*

Sets the actual spacing distance which did not meet requirements in a spacing violation. The option argument is ignored if the associated error type is not in the spacing class. The error class for the error type is set when it is created. The actual distance is used to provide a more detailed rendering of the spacing violation in the GUI layout view. The actual spacing value becomes available as the *actual_spacing* attribute for the error.

`-width_required` *`distance`*

Sets the required width of the drc_error object, which is the required width of the violating object. The required width has meaning in the context of a violation in the "size" error_class, where the width of the object does not meet the requirement.

`-height_required` *`distance`*

Sets the required height of the drc_error object, which is the required height of the violating object. The required height has meaning in the context of a violation in the "size" error_class, where the height of the object does not meet the requirement.

`-shape_uses` *`shape_uses`*

Sets the shape uses associated with the error. The *shape_uses* values can be used to filter errors in the GUI error browser where the attribute is called "route types". The *shape_uses* become available as the *shape_uses* attribute for the error.

`-pin_edge` *`pin_edge`*

Sets the pin edge associated with the error. Given any shape, rectangular or rectilinear shape, the edge numbers are a list of positive integers that start from 1. The lower left-most vertical edge is the starting edge with side number 1. Going clockwise, the side numbers of the next edges are 2, 3, and so on. The pin edge is reported as a part of the detailed textual report in the GUI error browser. The pin edge value becomes available as the *pin_edge* attribute for the error.

`-information` *information*

Sets the error instance specific information string attribute. Information for an error instance is generally set by setting the brief_info and verbose_info attributes of the owning error type and formatting error instance specific data into these strings. However, there are times that information specific to an error instance must be set. In this case, use this attribute to set the information string.

Description

The *create_drc_error* command creates a physical DRC error object in the given error data for the given error type. Additional shapes can be added to a drc_error object after creation by using the *create_drc_error_shapes* command.

Most of the attributes associated with a drc_error object are read-only and can only be set at the time of creation. Exceptions to this is the error status, which can be modified using the *set_attribute* command. All attribute values set at creation can then be accessed with the *get_attribute* command. In addition, the attribute values are displayed in the GUI error browser and the GUI layout view.

Brief and verbose information strings are accessible for each drc_error object as the *brief_info* and the *verbose_info* attributes, respectively. These strings are also displayed in the GUI error browser. These information strings are not attached to the drc_error object in the database, but rather with the drc_error_type associated with the drc_error object. A message construction service customizes the information strings for each drc_error object by substituting attribute values from the drc_error object into the strings. A detailed description of the message construction is provided in the man page for *create_drc_error_type*.

Examples

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then creates an error type in the spacing class. A DRC error instance is then created for the new type.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}

prompt> set type [create_drc_error_type -error_data $data \\
    -name "route spacing" -error_class spacing \\
    -required_objects {net net} \\
    -brief_info "A spacing violation between net %net:0% and net %net:1%
 was detected." \\
    -brief_format message \\
    -severity error]
{"route short"}

prompt> create_drc_error -error_data $data \\
    -error_type $type -objects [get_nets {"F_12_" "F_11_"}] \\
```

Feedback

```
    -polygons {{78.8600 189.4600} {79.8800 189.4600} {79.8800 186.3200}
 {78.8600 186.3200}} \\
    -status error -direction vertical \\
    -required_spacing 3.400 -actual_spacing 31.400
{"123"}
```

See Also

- create_drc_error_shapes
- create_drc_error_type
- open_drc_error_data
- remove_drc_errors

---

create_drc_error_data

Creates an error data file and returns a collection that contains the error data object.

Syntax

```
collection create_drc_error_data
[-file_name file_name]
-checker_name checker_name
[-checker_version version_string]
[-information info_string]
```

Data Types

```
file_name           string
checker_name        string
checker_version     string
information         string
```

Arguments

`-file_name file_name`

Specifies the name of the external error data file.

The naming convention for an error data file is to append an underscore, the checker name, and the ".err" suffix to the design name. For example, for a design named "my_design" and a checker named "test", the error data file name is named "my_design_test.err".

`-checker_name checker_name`

Specifies the name of the rule checking engine creating the error data file.

For example, for the route check performed by the router, the engine name might be "zroute".

`-checker_version` *version_string*

Specifies the version string for the checker engine.

`-information` *info_string*

Specifies an informative message to associate with the error data.

Description

The *create_drc_error_data* command creates the specified error data file. The command fails if the specified error data file already exists.

If created with the *-file_name* option, the error data file is written to disk. The created error data file is opened in read/write mode. To save an error data file created in this manner, run the *save_drc_error_data* command.

The command returns a collection that contains the error data object associated with the error data file. If no error data object is created, the command returns an empty string.

The *create_drc_error_data* command increments the open count for the error data object. The *create_drc_error_data* (and *open_drc_error_data*) commands must be balanced by an equal number of *close_drc_error_data* commands to close the error data object.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example creates an external error data file named "my_design_dppinassgn.err", with a checker name of "dppinassgn", and a short information string.

```
prompt> set data [create_drc_error_data \\
   -file_name my_design_dppinassgn.err \\
   -checker_name dppinassgn \\
   -information "This error data contains pin assignment and \\
      placement errors for the my_design design"]
{"my_design_dppinassgn.err"}
```

See Also

- close_drc_error_data
- get_drc_error_data
- open_drc_error_data
- remove_drc_error_data
- save_drc_error_data

Feedback

create_drc_error_shapes

Creates shapes and adds to a physical DRC error object.

Syntax

```
status create_drc_error_shapes
-error_data error_data
drc_error
[-polygons shape_list]
[-polylines shape_list]
[-endpoints coord_list]
[-points shape_list]
```

Data Types

```
error_data      collection
drc_error       collection
shape_list      list
coord_list      list
```

Arguments

`-error_data error_data`

Specifies the error data in which to find the physical DRC error to modify.

drc_error

Specifies the error to which to add the shapes.

`-polygons shape_list`

Polygon or rectangle shapes which describe the DRC error. The *shape_list* argument is a list of shapes where each shape is a list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the *polygons* attribute for the error.

`-polylines shape_list`

Describes polylines shapes for DRC error. The *shape_list* argument is list of shapes where each shapes consist of list of x- and y-coordinate pairs: {{*x y*} {*x y*} {*x y*} {*x y*} ... }. The shapes become available as the *polylines* attribute for the error.

`-endpoints coord_list`

Specifies the two endpoints of distance shape of violation such as spacing or open error. The *coord_list* argument is a list of two x- and y-coordinate pairs: { {*x y*} {*x y*} }. If more than two points are given, only the first two are used and the rest are ignored. This option is a required option when create open error. The coordinates become available as the *endpoints* attribute for the error.

Feedback

`-points` *shape_list*

Specifies points as shape of violation when necessary. The *shape_list* argument is a list of x- and y-coordinate pairs: { {*x y*} {*x y*} ... }. The coordinates become available as the *points* attribute for the error.

Description

The *create_drc_error_shapes* command adds additional shapes to an existing physical DRC error object in the given error data. Shapes are added to the DRC error object at the time of creation. If more than one shape is required to describe the error, use the *create_drc_error_shapes* command to add the additional shapes. The command requires at least one of shape options be specified.

Examples

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then creates an error type in the basic class. A DRC error instance is then created for the new type. Additional shapes are then added.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}

prompt> set type [create_drc_error_type -error_data $data \\
    -name "shapely error" -error_class basic \\
    -required_objects {net} \\
    -severity error]
{"route short"}

prompt> set error [create_drc_error -error_data $data \\
    -error_type $type -objects [get_nets {F_12_}] \\
    -polygons {{78.8600 189.4600} {79.8800 189.4600} {79.8800 186.3200}
 {78.8600 186.3200}}
{"45"}

prompt> create_drc_error_shapes -error_data $data $error\\
    -polygons {{74.8600 189.4600} {75.8800 189.4600} {75.8800 186.3200}
 {74.8600 186.3200}}
```

See Also

- create_drc_error
- create_drc_error_type
- open_drc_error_data

Feedback

create_drc_error_type

Creates a physical DRC error type object.

Syntax

```
collection create_drc_error_type
-error_data drc_error_data
-name type_name
-error_class base | spacing | short | open_locator
[-brief_info info_string]
[-brief_format ascii | message]
[-verbose_info info_string]
[-verbose_format ascii | message]
[-severity error | information | recommendation | warning]
[-required_objects {net pin port cell pinGuide}]
[-num_detected_errors num_errors]
```

Data Types

```
drc_error_data  collection
type_name       string
info_string     string
num_errors      integer
```

Arguments

`-error_data drc_error_data`

Specifies the error data in which to create the physical DRC error type.

`-name type_name`

The name for the error type being created. Names must be unique in the scope of the owning error data.

`-error_class base | spacing | short | open_locator`

Specifies the kind of errors represented by the new error type. The error class determines how the shapes of the associated DRC error instances are rendered in the GUI layout view. It also determines some expected attributes for its error instances. Available values are: base, spacing, short, and open_locator.

The *base* class allows its types to define required associated design objects. The *spacing* class allows its types to define required associated design objects and the error instances to describe a mismatch in actual and required spacing distance. Spacing errors can also be defined with a direction. The *short* class allows its types to define required associated design objects and is rendered with a diagonal cross inside of its shape. The *open_locator* class is for open nets and expects to be associated with a single net and to have two endpoints describing an open in a net as its shape description.

`-brief_info` *`info_string`*

Specifies a brief information string for display. Error instances of an error type are shown with brief and verbose information strings that are derived from the brief and verbose information strings associated with its error type. The strings can be customized for the error instance by using tokens that are replaced by the error instance's attribute values. See the DESCRIPTION section for a detailed explanation.

The brief information string is expected to be a compact description that is suitable for display in a tabular format in the GUI error browser.

`-brief_format ascii | message`

Specifies the display format of the brief information string specified with *-brief_info*. If the format is *ascii*, the brief info string is displayed exactly as specified with the *-brief_info* option, without modification. If the format is *message*, the brief info string is parsed for special tokens. The tokens are replaced with error instance attribute values before display.

`-verbose_info` *`info_string`*

Error instances of an error type are shown with brief and verbose information strings that are derived from the brief and verbose information strings associated with its error type. The strings can be customized for the error instance by using tokens that are replaced by the error instance's attribute values. See the DESCRIPTION section for a detailed explanation.

The verbose information string should be a detailed description that is suitable for display in a report format in the GUI error browser.

`-verbose_format ascii | message`

Specifies the display format of the verbose information string specified with *-verbose_info*. The format can be one of: *ascii* or *message*. If the format is *ascii*, the verbose info string is displayed exactly as specified with the *-verbose_info* option, without modification. If the format is *message*, then the verbose info string is parsed for special tokens. The tokens are replaced with error instance attribute value before display.

`-severity error | information | recommendation | warning`

Specifies the severity of the message associated with instances of this type.

`-required_objects {net pin port cell pinGuide}`

Specifies the required object classes associated with this error. A type might require that a specific number of design objects be associated with its error instances. For example, a route spacing type error might require that all of its instances have two associated nets. If the spacing violation is between shapes

of the same net, the same net can be repeated twice. By specifying required objects, the type allows instances to be filtered for null object association.

The required objects are specified with a list of the design object class names. If multiple instances of a design object type are required, the class name can be repeated. For example, to specify that two nets are required, use the option argument *{net net}*.

While this option allows definition of the required design object types, additional objects can be associated with an error instance in addition to the required design object types.

`-num_detected_errors` *num_errors*

Specifies the maximum number of detected errors that are allowed. Some checkers allow you to specify a limit on the number of errors detected per type. The checker can add information about how many errors were detected, vs. how many errors were added into the error database, by using this option.

Description

The *create_drc_error_type* command creates physical DRC error objects in the specified error data. Error types must have unique names in the scope of the owning error data and error types must be defined with an error class.

The command supports brief and verbose information strings for error type objects. These strings are displayed for each error instance in the GUI error browser. The strings can be customized with attribute values of an error instance by using the *message* format along with replacement of special tokens with attribute values.

A message construction service parses the error message for delimited tokens. If found, the command substitutes them with attribute values to construct a message appropriate for a specific DRC error instance. The token delimiter is the percent (%) character. To pass a percent symbol directly to the output message without modification, escape the character with a backslash: \\%. A token has the form *%<error instance attribute name[:<ordinal value>]>[.<attribute name>]%* Error instance attributes are named as described below.

For associated design objects, the attribute name is the object type name. If there are multiple objects of the same object type, the specific associated object is identified with the ":<ordinal value>" qualifier. The value starts at zero for the first identified object. For example "pin:0" identifies the first associated pin and "pin:2" identifies the third pin associated with the error instance. If the ordinal value qualifier is omitted, the first object of that type is default. The special character "*" can be used in place of the ordinal value to mean all objects of the given type. The supported object type names are: *net, pin, port, cell,* and *pinGuide*.

In addition, the attribute name *type* can be used to denote the associated error type. Append a period character (.) delimiter to the optional attribute name to access a specific

attribute for the identified object. If the attribute qualifier is omitted, the object's name is used. Attribute names supported with the associated object are: name and full_name. In addition, pin and port objects support the attribute name *cell* which will print the name of the cell associated with the pin or port. Attribute names supported with the associated error type are: name, severity, and class.

For other error instance attribute fields, access an attribute value by using the attribute name for the object. For example, use *actual* for the actual distance in a spacing violation and *direction* for the direction of a spacing violation. Some of these attributes can also be given with the *:<ordinal value>* qualifier. For example, there can be multiple route types associated with an error. As with design objects, if the ordinal value qualifier is omitted, the first one is selected by default. The * glob character can be used to specify all values. The currently supported error instance attributes are: *status, shapes, bbox, direction, actual, required, routeType,* and *endpoint*.

Examples

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then creates an error type in the short class.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> create_drc_error_type -error_data $data \\
    -name "route short" -error_class short \\
    -required_objects {net net} \\
    -brief_info "A short between net %net:0% and net %net:1% was
 detected." \\
    -brief_format message \\
    -severity error
{"route short"}
```

See Also

- create_drc_error
- open_drc_error_data
- remove_drc_error_types

---

create_generated_clock

Creates a generated clock object.

Syntax

string *create_generated_clock*

Feedback

```
[-name clock_name]
-source master_pin
[-divide_by divide_factor | -multiply_by multiply_factor |
 -edges edge_list]
[-combinational]
[-duty_cycle percent]
[-invert]
[-preinvert]
[-edge_shift edge_shift_list]
[-add]
[-master_clock clock]
[-pll_output output_pin]
[-pll_feedback feedback_pin]
[-sms_scenarios sms_scenarios_list]
[-comment comment_string]
source_objects
```

Data Types

clock_name	string
master_pin	list
divide_factor	int
multiply_factor	int
edge_list	list
percent	float
edge_shift_list	list
clock	string
output_pin	list
feedback_pin	list
sms_scenarios_list	collection
comment_string	string
source_objects	list

Arguments

`-name clock_name`

Specifies the name of the generated clock. If you do not use this option, the clock receives the same name as the first clock source specified in the *-source* option.

If you specify the *-add* option, you must use the *-name* option and the clocks with the same source must have different names.

`-source master_pin`

Specifies the master clock source pin. This pin is used to determine the following:

- The identity of the master clock
- The sense of the master clock (inverted or non-inverted) used as input for the generated clock

This pin does not restrict the generated clock's source latency computation; the tool considers all paths between the master clock's actual source pin and the generated clock that provide the required edge relationships. (This pin can even be outside the fanin of the generated clock's own source pin, as long as it provides the identity and sense of the desired master clock.)

If more than one potential master clock exists at the specified pin, use the *-master_clock* option to specify which clock to use.

`-divide_by` *`divide_factor`*

Specifies the frequency division factor. For example, *-divide_by 2* specifies that the generated clock period is twice as long as the master clock period.

`-multiply_by` *`multiply_factor`*

Specifies the frequency multiplication factor. For example, *-multiply_by 3* specifies that the generated clock period is one-third of the master clock period.

`-edges` *`edge_list`*

Specifies a list of integers that represents edges from the source clock that are to form the edges of the generated clock. The edges are interpreted as alternating rising and falling edges and each edge must be not less than its previous edge. The number of edges must be an odd number and not less than 3 to make one full clock cycle of the generated clock waveform. For example, *1* represents the first source edge, *2* represents the second source edge, and so on.

`-combinational`

Causes the source latency path for a *-divide_by 1* generated clock to include only the logic that is propagated from the master clock through combinational logic. The source latency path does not flow through sequential element clock pins, transparent latch data pins, or source pins of other generated clocks.

`-duty_cycle` *`percent`*

Specifies the duty cycle, in percent, if frequency multiplication is used. It is the percentage of the clock period that the clock is high. The default is 50 percent.

`-invert`

Inverts the generated clock signal (in the case of frequency multiplication and division). This option first creates the generated clock from the original clock signal and then inverts the generated clock.

`-preinvert`

Creates a generated clock based on the inverted clock signal. This option first inverts the original clock signal and then creates the generated clock signal based on the inverted original clock.

`-edge_shift` *`edge_shift_list`*

Specifies a list of floating-point numbers that represent the amount of shift, in library time units, of the specified edges to yield the final generated clock waveform. The number of edge shifts specified must be equal to the number of edges specified by the *-edges* option. The values can be positive or negative, representing shifts later or earlier in time, respectively. For example, the value 1.0 shifts the corresponding edge later by one library time unit. Use this option to specify shifts resulting from the logic of the design, not to model latency effects.

`-add`

Specifies whether to add this clock to the existing clock or to overwrite it. Use this option to capture the case where multiple generated clocks must be specified on the same source, because multiple clocks fan into the master pin. Ideally, one generated clock must be specified for each clock that fans into the master pin.

If you specify this option, you must also use the *-name* and *-master_clock* options.

Defining multiple clocks on the same source pin or port causes longer runtime and higher memory usage than a single clock, because PrimeTime must explore all possible combinations of launch and capture clocks. Use the *set_false_path* command to disable unwanted clock combinations.

`-master_clock` *`clock`*

Specifies the master clock to be used for this generated clock if multiple clocks fan into the master pin.

If you specify this option, you must also use the *-add* option.

`-pll_output` *`output_pin`*

Specifies the output pin of the PLL that is connected to the feedback pin. For a single-output PLL, this pin is same as the pin on which the generated clock is defined.

`-pll_feedback` *`feedback_pin`*

Specifies the feedback pin of the PLL. There should be a path in the circuit connecting one of the outputs of the PLL to this feedback pin.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this generated clock specification is analyzed. When this option is not given the generated clock specification applies for all SMS scenarios. PrimeTime requires clock waveforms to be specified for all applicable SMS scenarios. It is thus recommended to first specify the generated clock waveforms without this option, followed by the

desired waveforms at specific SMS scenarios using this option. This collection is created by *get_sms_scenarios*.

PrimeTime currently supports this option for only a single clock in a generated clock hierarchy. For example, if this option is used for a master clock, none of its generated clock children can use this option.

`-comment` *`comment_string`*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

`source_objects`

Specifies a list of ports, pins, or nets defined as generated clock source objects. When a net is used as the source, the first driver pin of the net is the actual source used to create the generated clock.

Description

The *create_generated_clock* command creates a generated clock object in the current design and also defines a list of objects as generated clock sources in the current design. You can specify a pin or a port as a generated clock source object. The command also specifies the original master clock source from which the new clock is generated. When the master clock changes, the generated clock automatically changes.

The generated clock can be created as one of the following:

- A frequency divided clock, using the *-divide_by* option
- A frequency multiplied clock, using the *-multiply_by* option
- A special divide-by-1 clock, using the *-combinational* option
- An edge-derived clock, using the *-edges* option

To invert a frequency-divided or frequency-multiplied clock, use the *-invert* or *-preinvert* option. To shift the edges of an edge-derived clock by specified amounts of time, use the *-edge_shift* option.

The number of edges specified by the *-edges* option must be an odd number of at least 3. Consider the following example:

```
create_generated_clock -source clk -edges {1 3 5} [get_pins U82/Q]
```

The first, third, and fifth edges of the source clock determine the times at which the first, second, and third edges (rising-falling-rising) occur for the generated clock. The period of the generated clock is determined by the final entry in the edge list.

Non-increasing edges as in *-edges { 1 1 3 }* are allowed and usually used with the *-edge_shift* option to produce a generated clock pulse independent of the duty cycle of the master clock itself.

If a generated clock is specified with a *divide_factor* value that is a power of 2 (1, 2, 4, ...), the rising edges of the master clock are used to determine the edges of the generated clock. If the *divide_factor* value is not a power of two, the edges are scaled from the master clock edges.

Using the *create_generated_clock* command on an existing generated_clock object overwrites the existing object attributes. The generated_clock objects are expanded to real clocks at the time of analysis.

The *set_clock_latency*, *set_clock_uncertainty*, *set_propagated_clock*, and *set_clock_transition* commands can reference the generated clock.

For internally generated clocks, the tool automatically computes the clock source latency if the master clock of the generated clock has propagated latency and no user-specified value for generated clock source latency exists. If the master clock is ideal and has source latency and there is no user-specified value for the generated clock's source latency, zero source latency is assumed.

To display information about generated clocks, use the *report_clock* command.

Examples

The following example creates a frequency divide-by 2 generated clock.

```
pt_shell> create_generated_clock -divide_by 2 \\
            -source [get_pins CLK] [get_pins pinf]
```

The following example creates a frequency divide-by 3 generated clock. If the master clock period is 30, and master waveform is {24 36}, the generated clock period is 90 with waveform {72 108}.

```
pt_shell> create_generated_clock -divide_by 3 \\
            -source [get_pins CLK] [get_pins div3/Q]
```

The following example creates a frequency multiply-by 2 generated-clock with a duty cycle of 60%.

```
pt_shell> create_generated_clock -multiply_by 2 -duty_cycle 60 \\
            -source [get_pins CLK] [get_pins pinf1]
```

The following example creates a frequency multiply-by 3 generated-clock with a duty cycle equal to the master clock duty cycle. If the master clock period is 30, and master waveform is {24 36}, the generated clock period will be 10 with waveform {8 12}.

```
pt_shell> create_generated_clock -multiply_by 3 \\
            -source [get_pins CLK] [get_pins div3/Q]
```

The following example creates a generated clock with edges at first, third, and fifth edges of the master clock source.

```
pt_shell> create_generated_clock -edges {1 3 5} \\
            -source [get_pins CLK] [get_pins pinf2]
```

The following example creates a generated clock like the previous example, but with each derived edge shifted later by one time unit.

```
pt_shell> create_generated_clock -edges {1 3 5} -edge_shift {1.0 1.0 1.0}
 \\
            -source [get_pins CLK] [get_pins pinf2]
```

The following example creates a generated clock that is inverted with respect to the source clock and has the same period.

```
pt_shell> create_generated_clock -divide_by 1 -combinational -invert \\
            -source [get_pins CLK] [get_pins pinf2]
```

The following example creates a rising-pulse clock with a width of 5.0 time units, with the pulse triggered by the rising edge of the master clock, and the period determined by the third edge of the master clock.

```
pt_shell> create_generated_clock -edges {1 1 3} -edge_shift {0 5.0 0}  \\
            -source [get_pins CLK] [get_pins pinf2]
```

The following example creates a falling-pulse clock triggered by the rising edge of its master clock.

```
pt_shell> create_generated_clock -edges {1 1 3} -edge_shift {0 5.0 0} \\
            -invert -source [get_pins CLK] [get_pins pinf2]
```

The following example creates a frequency doubling pulse. A rising pulse is triggered by both rising and falling edges of its master clock.

```
pt_shell> create_generated_clock -edges {1 1 2 2 3} -edge_shift {0 2.5 0
 2.5 0} \\
            -source [get_pins CLK] [get_pins pinf2]
```

See Also

- check_timing
- create_clock
- get_generated_clocks
- remove_generated_clock
- report_clock
- set_clock_latency

Feedback

- set_clock_transition
- set_clock_uncertainty
- set_false_path
- set_propagated_clock
- timing_gclock_expand_odd_period_edge_list

---

create_histogram

Creates a custom histogram graph.

Syntax

status *create_histogram*

```
[-numbins number_of_bins]
[-min min_value]
[-max max_value]
[-greater_than gt_value]
[-less_than lt_value]
[-midpoint mid_value]
[-worst worst_side]
[-title title_label]
[-xlabel x_axis_label]
[-ylabel y_axis_label]
[-object_name object_name]
[-value_label value_label]
[-attribute attribute_name]
[-tcl_cmd tcl_command]
[-name name_histogram]
[-view view_name]
[-width window_width]
[-height window_height]
[-bin_range bin_range]
[-max_num_bins max_num_bins]
[-grid grid_view]
-collection object_list
[-infile debug_input]
[-outfile debug_output]
[-core_infile debug_core_input]
[-core_outfile debug_core_output]
```

Data Types

number_of_bins	integer
min_value	float
max_value	float
gt_value	float
lt_value	float

```
mid_value           float
worst_side          string
title_label         string
x_axis_label        string
y_axis_label        string
object_name         string
value_label         string
attribute_name      string
tcl_command         string
name_histogram      string
view_name           string
window_width        integer
window_height       integer
bin_range           integer
max_num_bins        integer
grid_view           string
object_list         list
debug_input         string
debug_output        string
debug_core_input    string
debug_core_output   string
```

Arguments

`-numbins` *number_of_bins*

An integer in the range of 1 to 300, that specifies the number of bins to use in creating the histogram. The default is 8.

`-min` *min_value*

Specifies the minimum value to use in binning objects; the default is the smallest value used in binning the objects, or the value of *-greater_than*, if specified. Any object whose associated value is less than this value is to be dropped from the histogram with a warning message.

`-max` *max_value*

Specifies the maximum value to use in binning objects; the default is the largest value used in binning the objects, or the value of *-less_than*, if specified. Any object whose associated value is greater than this value is to be dropped from the histogram with a warning message.

`-greater_than` *gt_value*

Specifies the non-inclusive least value limit to use in binning objects; by default, there is no filtering of data on the min end of the value range. If *-min* is specified and *-greater_than* is not, the value given for *-min* is used. Any object whose associated value is less than or equal to this value is to be dropped from the histogram with a warning message.

`-less_than` *`lt_value`*

Specifies the non-inclusive greatest value limit to use in binning objects; by default, there is no filtering of data on the max end of the value range. If *-max* is specified and *-less_than* is not, the value given for *-max* is used. Any object whose associated value is greater than or equal to this value is to be dropped from the histogram with a warning message.

`-midpoint` *`mid_value`*

Specifies the midpoint value to use in interpreting histogram bins; by default, no midpoint is used. An x-axis tick mark is placed on the midpoint value and all bins to the *worst* side of the midpoint are colored red while all bins to the other side of the midpoint are colored green. The *worst* side is *none* by default but can be set to *left* or *right* using the *-worst* option.

`-worst` *`worst_side`*

Specifies whether values less than or greater than the midpoint are to be considered "worst." Allowed values are *left*, meaning that values less than midpoint are worse; *right*, meaning that values greater than the midpoint are worse; and *none* (the default), meaning that neither is worse.

`-title` *`title_label`*

Specifies the label to place at the top of the histogram graph. If not specified, a default title is constructed by concatenating the *object_name* label and the *value_label* label.

`-xlabel` *`x_axis_label`*

Specifies the label to place along the x-axis of the histogram graph. If not specified, the default x-axis label is the *value_label* label.

`-ylabel` *`y_axis_label`*

Specifies the label to place along the y-axis of the histogram graph. If not specified, the default y-axis label is constructed with the string "Number of *object_name*" where *object_name* is replaced with the *object_name* label.

`-object_name` *`object_name`*

Specifies the label to place at the head of the object name column in the list view accompanying the histogram. If not specified, the default is the standard type name for the type of objects found in the object list supplied for the *-collection* argument. For example, if the list consists of pins, the label "Pin" is placed at the head of the object name column. If the object list consists of multiple types of objects, the label "Object" is used. This label can also be used to construct the default title and y-axis labels.

`-value_label` *`value_label`*

Specifies the label to place at the head of the value column in the list view accompanying the histogram. If not specified, the default is the attribute name specified for the *-attribute* option, or the Tcl command string specified for the *-tcl_cmd* option, whichever is used. This label can also be used to construct the default title and x-axis labels.

`-attribute` *`attribute_name`*

Specifies the name of the attribute to use in constructing the histogram. The value of the named attribute is fetched for each object in the object list specified by the *-collection* argument. These values are then used to bin the objects to construct the histogram. If the attribute value is not defined for any object in the list, a warning message is issued and the object is excluded from the histogram. You must specify either *-attribute* or *-tcl_cmd*, but not both.

`-tcl_cmd` *`tcl_command`*

Specifies the Tcl command to use in constructing the histogram. The Tcl command is evaluated for each object in the object list specified by the *-collection* argument. These values are then used to bin the objects to construct the histogram. If the Tcl command evaluation fails for any object in the list, a warning message is issued and the object is excluded from the histogram. You must specify either *-attribute* or *-tcl_cmd*, but not both.

For a description of how the object in the object list is passed to the Tcl command, see the DESCRIPTION section.

`-name` *`name_histogram`*

Specifies the name of the new histogram view.

`-view` *`view_name`*

Specifies an existing histogram view to use for the new histogram.

`-width` *`window_width`*

Specifies the width of the window in number of pixels. The default is the width of the last histogram window closed when PrimeTime was last run.

`-height` *`window_height`*

Specifies the height of the window in number of pixels. The default is the height of the last histogram window closed when PrimeTime was last run.

`-bin_range` *`bin_range`*

Specifies the size of the value range per bin.

`-max_num_bins` *`max_num_bins`*

Specifies the maximum number of bins allowed when value range per bin is specified. The value must be greater than or equal to 1.

`-grid` *`grid_view`*

Specifies an existing grid view to use with the new histogram.

`-collection` *`object_list`*

Specifies the object collection to bin and graph.

`-infile` *`debug_input`*

Specifies the file to read data from on the GUI side.

`-outfile` *`debug_output`*

Specifies the file to write data to on the GUI side.

`-core_infile` *`debug_core_input`*

Specifies the file to read data from on the core side.

`-core_outfile` *`debug_core_output`*

Specifies the file to write data to on the core side.

Description

The *create_histogram* command constructs a histogram by binning the objects in the given object list using values evaluated with the given attribute name or Tcl command. You can issue *create_histogram* only while the GUI is active. Type the command into the command line field at the bottom of the GUI console window.

The object in the object list is passed to the Tcl command using the following conventions.

For each object the following is appended to the supplied Tcl command string: "-collection *collection_name* -index *index_of_object*" where *collection_name* is replaced by the name of the list given to the *create_histogram* command and the *index_of_object* is replaced with the zero-based index of the current object under evaluation in that list.

If the object is not a timing path, timing arc, or library timing arc, the following is further appended after the *-collection* and *-index* arguments: "-object *object*" where *object* is replaced with the name of a one-object collection that contains the current object under evaluation from the object list.

Note that you cannot use the *index_collection* command as the *-tcl_cmd* argument for lists of timing paths, timing arcs, library timing arcs, and timing points.

The histogram is graphed in a histogram view accompanied by a two-column table view that displays members of a selected bar. You can select a bar in the histogram by clicking the left mouse button over the bar. When a bar is selected, the accompanying table view

displays the objects in the bar. The left column displays the object names, while the right column displays the values associated with the objects used to created the histogram.

Examples

These examples show creation of custom histograms using the *create_histogram* command. All command strings must be entered in the command line field at the bottom of the GUI console window.

The following example shows creation of a histogram using the *-attribute* flag to retrieve an attribute value for each object in a collection. In this example, the *fanout_load* attribute value is used to bin all pins in the design. The graph is labeled with the title string "Pin Fanout" using the *-title* option. A new window frame is created for the new histogram by default because no view is given with a *-view* option. The new histogram view is named *view1* using the *-name* option,

```
pt_shell> create_histogram -name view1 -attribute fanout_load \\
                -title "Pin Fanout" -collection [get_pins *]
1
```

The following example shows creation of another histogram binning cells by area, with the graph reusing the histogram view named *view1* created in the previous example. The histogram will have the default title "Cell area", default x-axis label "area" and y-axis label "Number of cells".

```
pt_shell> create_histogram -view view1 -attribute area \\
                -collection [get_cells *]
1
```

The following example shows creation of a histograms using the *-tcl_cmd* flag to use a user-defined Tcl procedure to retrieve a value for each object in a collection. First, a Tcl script file named "myscript.tcl" containing the following lines is created to define a Tcl procedure named "myproc" that retrieves the *fanout_load* attribute value for the object that is passed in the procedure arguments:

```
proc myproc {flag1 clct flag2 idx flag3 obj} {
    set result [get_attribute $obj fanout_load]
    return $result
}
```

The disregarded strings flag1, flag2, and flag3 are placeholders for the *-collection*, *-index*, and *-object* option flags, respectively, preceding the three arguments.

You can identify the object by directly using the object in the third argument as in the preceding myproc procedure, or by using the collection name and the object index in the first and second arguments, respectively. The following procedure, myproc2, performs the same function as myproc but retrieves its objects using the first and second arguments:

```
proc myproc2 {flag1 clct flag2 idx flag3 obj} {
    set mypin [index_collection $clct $idx]
```

```
    set result [get_attribute $mypin fanout_load]
    return $result
}
```

You can now source the script file to define the procedures within the PrimeTime Tcl interpreter context:

```
pt_shell> source myscript.tcl
1
```

After you define the procedures, you can reference them by running *create_histogram -tcl_cmd* as in the following example.

```
pt_shell> create_histogram -name view1 -tcl_cmd myproc \\
          -title "Pin Fanout" -collection [get_pins *]
1
```

In this example, the embedded command *[get_pins *]* is first interpreted to create a collection of all pins in the design. The Tcl procedure "myproc" is then invoked for each pin object in the collection to retrieve a value per pin to use in creating the histogram.

See Also

- collections

---

create_ilm

Creates an interface logic model and writes it to a new directory. Also, sets the *is_interface_logic_pin* attribute on pins of the current design that are part of its interface logic model.

Syntax

int *create_ilm*

```
[-script_format ptsh | dcsh | dctcl]
[-instances instance_list]
[-ignore_ports port_list]
[-auto_ignore]
[-verbose]
[-ignore_boundary_pins boundary_pin_list]
[-include incl_list]
[-verification_script]
[-latch_level levels]
[-context_borrow]
[-keep_ignored_fanout]
[-include_pins pin_list]
[-critical_pins]
[-traverse_disabled_arcs]
[-parasitics_options para_options]
```

```
[-sdf_options sdf_options]
[-validate valid_list]
```

Data Types

```
instance_list       list
port_list           list
boundary_pin_list   list
incl_list           list
levels              int
pin_list            list
para_options        list
sdf_options         list
scenario_name       string
valid_list          string
```

Arguments

`-script_format ptsh | dcsh | dctcl`

Specifies the output format for the script. Allowed values are *ptsh* (the default) for pt_shell, *dcsh* for dc_shell, and *dctcl* for dc_shell-t.

`-instances instance_list`

Specifies the list of instances for which ILMs need to be generated. Separate ILMs are generated for each instance and written out to directories named after the instance name.

`-ignore_ports port_list`

Specifies a list of input or output ports whose fanout or fanin is to be ignored when setting the *is_interface_logic_pin* attribute. Clock ports are automatically ignored; you do not have to specify them in this list.

You can use this option to exclude the fanout of ports connected to chip-level nets (for example, scan enable and reset) from affecting the contents of interface logic. If these nets are not explicitly ignored, they cause unnecessary logic (for example, internal registers) to be part of the interface logic on the current design. You can also use this option to selectively generate the interface logic for a subset of the ports on a block; for example, an interface logic model (ILM) can model only the timing behavior on a subset of the ports on a block.

By default, all nonclock input and all output ports are used in identifying the contents of interface logic. The *-ignore_ports*, *-ignore_boundary_pins*, and *-auto_ignore* options are mutually exclusive.

`-auto_ignore`

Enables automatic determination of ports whose fanout should be ignored when setting the *is_interface_logic_pin* attribute. A port is ignored if the percentage of the total registers in the design in the transitive fanout of the port exceeds a

specified threshold percentage contained in the *ilm_ignore_percentage* variable (default 25).

You can use this option to help you identify the test enable and reset ports of your design. Before using it, examine the current value of the *ilm_ignore_percentage* variable and reset it, if necessary. Note that the *-auto_ignore* option might potentially ignore ports you do not want to ignore, or fail to ignore ports you want to ignore. Carefully read the messages issued by this command when you use this option to see which ports have been ignored and what percentage of registers to which they fanned out.

The *-ignore_ports*, *-ignore_boundary_pins* and *-auto_ignore* options are mutually exclusive.

`-verbose`

Indicates that information about the number of cells and nets in the original design and in the ILM netlist is to be written to standard output.

`-ignore_boundary_pins` *`boundary_pin_list`*

Specifies a list of boundary pins whose fanout or fanin is to be ignored when placing the is_interface_logic_pin attribute while extracting an instance ILM. Works similar to the *-ignore_ports* option but works for instance ILMs. This option can only be used along with the *-instances* option.

The *-ignore_ports*, *-ignore_boundary_pins* and *-auto_ignore* options are mutually exclusive.

`-include` *`incl_list`*

Includes specified elements in the ILM. You can specify these values:

- *net_pins* - Includes all pins on nonclock interface logic nets and propagated clock interface logic nets. Use this option if the interface logic model (ILM) is used in non-SDF flows and delay calculation is performed using the information contained in the ILM. Preserving all pins on a net maintains correct pin capacitance information for the net. The *net_pins* does not affect unpropagated clock nets; that is, nets in the clock network that have user-specified source latency.
- *boundary_cells* - Includes the boundary cells for all input ports. By default, the boundary cells would be present for all inputs except for the ignored ports. This option includes boundary cells for the ignored input ports also so that DRC checks can be performed at top level.

- *si_delay_pins* - Includes additional logic required to perform crosstalk delay analysis at the chip-level. Since the crosstalk flow involves detailed parasitics, the *si_delay_pins* argument also turns the *net_pins* argument on.
- *si_noise_pins* - Includes additional logic required to perform noise analysis at the chip-level. Since the noise flow involves detailed parasitics, the *si_noise_pins* argument also turns the *net_pins* argument on.

`-verification_script`

Specifies that a script needs to be generated that can be used for the verification of ILM. By default, *create_ilm* command generates a script that can be used when the ILM is instantiated at upper level of hierarchy. This option additionally generates a script that can be used to validate the ILM against the original block from which the ILM was generated.

`-latch_level` *`levels`*

Specifies the number of logic levels over which time borrowing can occur for latch chains that are a part of the interface logic. Substitute the number of levels you want for the *levels* option. By default, all latches found in interface logic are assumed to be potential borrowers. Thus, all logic from I/O ports to flip-flops or output ports are identified as belonging to interface logic.

When you use this option, note that the value of the *levels* argument must account for the borrowing behavior of latches only on interface timing paths. If *n* represents a latch level, then *n* + 1 represents the number of latches maintained in latch chains that originate at input ports. The *n* + *1*th latch functions as an edge-triggered register and decouples I/O paths from internal paths. Use this option only when there are latches in the interface logic for a block. Specifying this option might potentially result in less accurate models, because not all latches are allowed to borrow. The *-context_borrow* and *-latch_level* options are mutually exclusive.

`-context_borrow`

Specifies that latch borrowing at the interface should be established based on the current context defined for the design. So, latches that borrow based on arrival times defined on input ports and clocks defined on the design are traced through, but path tracing stops at nonborrowing latches. The *-context_borrow* and *-latch_level* options are mutually exclusive.

`-keep_ignored_fanout`

Specifies that the fanout from ignored input ports to interface logic is to be maintained in the model. Thus, ignored ports are not used to identify interface logic, but connections between ignored ports and interface logic are preserved. Timing slacks for ignored ports might differ from those reported on the original

netlist because connections from ignored ports to internal registers are not preserved in the model.

`-include_pins` *`pin_list`*

Specifies a list of pins that must be included in the interface logic. Substitute the list of pins you want for the *pin_list* option. This option can be used to optionally add internal pins to the interface logic. After the interface logic is determined, this list of pins is added to the interface logic. Use this option to define additional interface logic pins that are otherwise removed in the model. There is not any affect on pins that are already in the interface logic. Use this option in conjunction with the *all_fanin* and *all_fanout* commands to include a particular cone of logic in the model.

`-critical_pins`

Specifies that critical pins interface logic must be identified. This option allows you to extract a model that keeps only the pins in the critical paths of the design.

You can use this option to extract a critical pins interface logic. The model generated contains the interface logic pins in the critical paths of the design. The model is compact compared to normal interface logic models, but might not be as accurate outside of the current context. This model can be used only to do critical path analysis. It might take much longer to generate this model than the normal model.

The *-include {si_delay_pins}* option is not currently supported for critical pins ILM.

`-traverse_disabled_arcs`

Specifies that the interface logic should not be affected by disabled arcs/pins on the design. If specified, this option forces the traversal of disabled arcs while determining interface logic.

`-parasitics_options` *`para_options`*

Writes parasitic data for the net pins of the ILM in addition to the following options:

- *spef_format* - Writes parasitic data in SPEF format.
- *sbpf_format* - Writes parasitic data in SBPF format.
- *input_port_nets* - Writes parasitic data for nets connected to the input ports on the current design.
- *constant_nets* - Writes parasitic data for constant nets.

`-sdf_options` *`sdf_options`*

Writes SDF data for the ILM with the following options:

- *annotated* - Includes only timing arcs that have been annotated with the *read_sdf*, *set_annotated_delay*, or *set_annotated_check* commands.
- *no_edge* - Omits any edges (posedge or negedge) for combinational I/O paths.
- *2.1_version* - Writes the output as SDF version 2.1. This is the default.
- *3.0_version* - Writes the output as SDF version 3.0.
- *input_port_nets* - Includes delays of nets connected to input ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available.
- *output_port_nets* - Includes delays of nets connected to output ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available.

`-validate` *`valid_list`*

Specifies what should be validated after the model is created. The allowed value is *timing*.

Description

Extracts an interface timing model (ILM) for the design or given list of instances. The ILM is written to a directory with name of the design, or the hierarchical name of instance with forward slash (/) replaced by underscore (_). The location of this potentially new directory is given by the *pt_ilm_dir* variable.

The command generates the following files:

- *dir_name*/ilm.v (Verilog design for the ILM)
- *dir_name*/ilm_inst.pt.gz (script to apply the instance constraints)

Additionally, the command might generate:

- *dir_name*/ilm.{sbpf, spef.gz} (Parasitics if *-parasitics_options* is given)
- *dir_name*/ilm.sdf (SDF file if *-sdf_options* is given)
- *dir_name*/ilm_verif.pt.gz (verification script for the design)
- *dir_name*/ilm.txt (list of aggressor annotation pins if *-include {si_delay_pins}* is given)

Also, the command sets the *is_interface_logic_pin* attribute on pins of the current design that are part of its interface logic.

The interface logic on a block contains all cells whose timing is affected by, or affects, the external environment of a block:

- All cells in timing paths that lead from input ports to registers or output ports that terminate the paths.
- All cells in timing paths that lead to output ports from registers or input ports that originate the paths.
- Clock trees that drive interface registers; including any registers in the clock tree. Clock-gating circuitry is part of interface logic if it is driven by external ports, but not if it is driven by registered outputs on a block.

Notice that interface logic does not include internal register-to-register paths and logic on a block associated only with these paths.

To include additional logic, use the *-include_pins* option. If the *-include {si_delay_pins}* option is given, additional register-to-register logic can be pulled in that is needed for crosstalk analysis at chip level.

This command implicitly performs an update_timing on the design if required. You can review the objects you have identified as interface logic by using the *get_ilm_objects* command.

If the *-validate* option is given, and the *hier_modeling_version* variable is set to 2.0, the model is validated automatically after the model is created.

Examples

The following example extracts ILM and sets the *is_interface_logic_pin* attribute on all nonclock pins in the current design, except for pins in the fanin/fanout of ports port1, port2, and port3.

```
pt_shell> create_ilm -ignore_ports [get_ports {port1 port2 port3}]
```

The following example extracts ILM for instance I1, additionally includes three levels of logic from internal pin I1/ff/Q and also includes internal crosstalk pins needed for top-level crosstalk analysis.

```
pt_shell> set ilm_pins [all_fanout -level 3 \\
          -flat [get_pin I1/ff/Q]]

pt_shell> create_ilm -instances {I1} \\
          -include_pins $ilm_pins -include {si_delay_pins}
```

The following example extracts a critical pins interface timing model.

```
pt_shell> create_ilm -critical_pins -include {net_pins}
```

The following example extracts ILM by placing the *is_interface_logic_pin* attribute on all nonclock pins in the current design, except for pins identified by the *-auto_ignore* option.

Feedback

Because the value of the *ilm_ignore_percentage* variable is currently 35, pins are ignored if the percentage of the total design registers in their transitive fanout is greater than 35. The *-latch_level* option with a value of 1 states that the first latch encountered in I/O paths might potentially borrow, but the second latch can be treated as an edge-triggered device. Thus, two levels of latches are maintained in the interface logic.

```
pt_shell> printvar ilm_ignore_percentage
ilm_ignore_percentage = "35"

pt_shell> create_ilm -auto_ignore -latch_level 1
```

See Also

- hier_modeling_version
- ilm_ignore_percentage
- pt_ilm_dir

---

create_merged_modes

Merges modes from the defined scenarios so that the timing scenarios are reduced.

Syntax

status *create_merged_modes*

```
[-test_only]
[-interactive]
[-keep_all_clocks]
[-match_clock_names]
[-target_eco]
[-value_tolerance tolerance]
```

Data Types

tolerance	float

Arguments

`-test_only`

Generates a mode merging report on which modes are merged and why certain modes are not be merged, without performing the actual merging.

`-interactive`

Determines which modes are merged and why certain modes are not merged by doing *-test_only* analysis, and opens the report in a GUI for interactive debugging.

`-keep_all_clocks`

Keeps all clocks separately. If you do not specify this option, mode merging adds only one clock into the merged mode when identical clocks (with the same waveform and sources) are present in more than one individual mode.

`-match_clock_names`

Specifies that clock names should also be matched in addition to the waveform and sources. If you do not specify this option, mode merging adds only one clock into the merged mode when identical clocks (with the same waveform and sources) are present in more than one individual mode.

`-target_eco`

Specifies that the merged modes are targeted for use in ECO flow. If you specify this option, mode merging automatically resolves merge conflicts wherever possible to improve mergeability. In addition, only the clocks that contribute to worst slack in the merged mode are kept and all other clocks are pruned.

`-value_tolerance` *tolerance*

When two modes have constraint values (for example, *set_clock_uncertainty*, *set_input_delay*, and so on) that differ by more than certain fraction, the two modes are not merged. By default, a tolerance value of 0.01 is used, which means that if the values are different by more than 1 percent, then the modes are not merged. You can use this option to change the tolerance value, thereby allowing for tradeoff between level of mode reduction and pessimism. A higher (or lower) tolerance value leads to more (or less) reduction but more (or less) pessimism in merged constraints.

Description

The *create_merged_modes* command merges modes from the defined scenarios so that the timing scenarios are reduced. For mode merging, the scenarios must have been created previously using *-mode* and *-corner* options of *create_scenario*. This command is only available in multi-scenario analysis.

After the modes are merged, the merged modes should be used with the same *-corner* option as was used for individual modes. Merged mode constraints are created under *multi_scenario_working_directory* with the name *merged_constraints*. Constraints are also written for unmerged modes in the directory. In addition, details of mode merging results, including why certain modes could not be merged, are written to a text file named *mode_merging_report.txt* in the *merged_constraints* directory.

Next, you describe how various commands are dealt with for mode merging. Mode merging merges pure mode commands (such as *create_clock*) and scenario (combination of mode and corner) commands. For scenario commands (such as *set_input_delay*), it is assumed that the command is present in all corners of a given mode but the values can

differ in each mode. For pure corner commands, it assumes that the command is present in all modes of the same corner.

Mode merging handles the following pure mode and scenario commands:

- create_clock • create_generated_clock • group_path • set_annotated_transition • set_case_analysis • set_clock_gating_check • set_clock_groups • set_clock_latency • set_clock_sense • set_clock_transition • set_clock_uncertainty • set_disable_timing • set_drive_resistance • set_driving_cell • set_false_path • set_fanout_load • set_ideal_latency • set_ideal_network • set_ideal_transition • set_input_delay • set_input_transition • set_load • set_max_capacitance • set_max_delay • set_max_fanout • set_max_time_borrow • set_max_transition • set_min_delay • set_mode • set_multicycle_path • set_output_delay • set_propagated_clock • set_sense

Although mode merging does not merge the following pure corner commands, the tool captures and writes these commands to merged mode scripts:

- create_operating_conditions • set_annotated_check • set_annotated_clock_network_power • set_annotated_delay • set_annotated_power • set_aocvm_coefficient • set_aocvm_table_group • set_connection_class • set_coupling_separation • set_cross_voltage_domain_analysis_guardband • set_domain_supply_net • set_dont_override • set_dont_touch • set_dont_touch_network • set_equal • set_input_noise • set_isolation • set_isolation_control • set_level_shifter_strategy • set_level_shifter_threshold • set_lib_rail_connection • set_library_driver_waveform • set_max_area • set_min_capacitance • set_min_pulse_width • set_noise_derate • set_noise_immunity_curve • set_noise_lib_pin • set_noise_margin • set_noise_parameters • set_operating_conditions • set_opposite • set_port_fanout_number • set_power_derate • set_rail_voltage • set_related_supply_net • set_resistance • set_retention • set_retention_control • set_setup_hold_pessimism_reduction • set_si_aggressor_exclusion • set_si_delay_analysis • set_si_noise_analysis • set_si_noise_disable_statistical • set_steady_state_resistance • set_supply_net_probability • set_switching_activity • set_temperature • set_timing_derate • set_voltage • set_wire_load_min_block_size • set_wire_load_mode • set_wire_load_model • set_wire_load_selection_group

For pure corner commands, any nested command is replaced by its output. For example, if the input constraints have the command "set_resistance 0.5 [get_nets n1]", the merged constraints have "set_resistance 0.5 {n1}". If net n1 does not exist, the merged constraints have "set_resistance 0.5", which is invalid syntax, leading to an error. It recommended that you check for and fix any errors in the input constraints on the pure corner commands.

The following list shows scenario commands that might need user intervention. Check these commands for correctness. For example, some clock names might need to be changed based on MMODE-008 messages.

Feedback

- set_power_clock_scaling • set_pulse_clock_max_transition • set_pulse_clock_max_width • set_pulse_clock_min_transition • set_pulse_clock_min_width

The following commands are not supported in mode merging.

- read_ddc • read_milkyway • set_active_clocks

Examples

In the following example, an iterative loop creates a set of scenarios. The same single.tcl script is used for all scenarios but the corner variables distinguishes the scenarios. Finally, the modes are merged using the *create_merged_modes* command. Modes M1 and M2 are merged to the new mode merged_1. Mode M3 is not merged.

```
pt_shell> foreach corner {bc tc wc} {
  foreach mode {M1 M2 M3} {
    create_scenario \
     -mode ${mode} \
     -corner ${corner} \
     -common_variables {mode corner} \
     -common_data "single.tcl"
  }
}

pt_shell> create_merged_modes
 ...
          Merged Mode Generation Summary
Original Modes              Merged Mode    Corners
-----------------------------------------------------
M1, M2                      merged_1       bc, tc, wc
M3                          M3             bc, tc, wc
```

See Also

- create_scenario

---

create_net

Creates nets in the current design.

Syntax

status *create_net*

```
[-exact]
net_list
```

Data Types

```
net_list     list
```

Feedback

Arguments

`-exact`

Indicates that names are to be used exactly as specified. Use this option if you want to create a net that contains a hierarchy or wildcard character as part of the net name. For more information, see "Naming Nets with Special Characters".

`net_list`

Specifies a list of nets to be created.

Description

The *create_net* command creates new nets in the current design. Like all other netlist editing commands, for the *create_net* command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The *create_net* command returns a 1 if successful and a 0 if unsuccessful.

The *create_net* command behaves almost identically to the *create_cell* command. Nets are created in scope; that is, at or below the current instance. Net names are specified as for other commands, using the full hierarchical name relative to the current instance, as in the following example:

```
create_net u1/u2/new_net class/AN2
```

This command attempts to create a net named new_net in the hierarchical block u1/u2. The name to the right of the last hierarchical separator is the actual net name, and the remainder is sent to the search engine to find a hierarchical block in which to create the new net. The operation fails if any of the following are true:

- The search engine cannot find "u1/u2".
- The search engine finds multiple blocks that match "u1/u2".
- The search engine finds a leaf cell matching "u1/u2".
- The search engine finds a net, port or hierarchical pin matching "u1/u2/new_net".

Naming Nets With Special Characters

If you want to create a net whose name contains the current hierarchical separator or wildcard characters used by the search engine, you must do the following:

- Use *current_instance* to change the scope to the block in which you want the new net created.
- Use *create_net -exact* to create the net.

For examples, see the EXAMPLES section.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Feedback

Examples

The following example creates nets in the current instance, and at a level below the current instance. Currently, u1 is an instance of a design that has multiple instances. Therefore, it is uniquified as part of the editing operation.

```
pt_shell> create_net new_net1
Information: Created net 'new_net1' in design 'top'. (NED-016)
1

pt_shell> create_net u1/new_net1
Uniquifying 'u1' (block1) as 'block1_0'.
Information: Created net 'new_net1' in 'top/u1'. (NED-016)
1
```

The following example creates net a/b in the hierarchical block u1/u2. The first attempt fails, because the *-exact* option was omitted. Here, u1/u2 is an instance of a design that has multiple instances; therefore, it is uniquified as part of the editing operation. u1 was already uniquified in the previous example.

```
pt_shell> current_instance u1/u2
u1/u2

pt_shell> create_net a/b
Error: Could not create net 'a/b':
        a not found. (NED-041)
Error: No changes made. (NED-040)
0

pt_shell> create_net a/b -exact
Uniquifying 'u1/u2' (block2) as 'block2_0'.
Information: Created net 'a/b' in 'top/u1/u2'. (NED-016)
1
```

See Also

- create_cell
- remove_net
- size_cell
- swap_cell
- write_changes

---

create_operating_conditions

Creates a new set of operating conditions in a library.

Feedback

Syntax

int *create_operating_conditions*

```
-name name
-library library_name
-process process_value
-temperature temperature_value
-voltage voltage_value
[-tree_type tree_type]
[-calc_mode calc_mode]
[-rail_voltages rail_value_pairs]
```

Data Types

name	string
library_name	string
process_value	float
temperature_value	float
voltage_value	float
tree_type	string
calc_mode	string
rail_value_pairs	list

Arguments

`-name` *name*

Specifies the name of the new set of operating conditions.

`-library` *library_name*

Specifies the name of the library for the new operating conditions.

`-process` *process_value*

Specifies the process scaling factor for the operating conditions. Allowed values are 0.0 through 100.0.

`-temperature` *temperature_value*

Specifies the temperature value, in degrees Celsius, for the operating conditions. Allowed values are -300.0 through +500.0.

`-voltage` *voltage_value*

Specifies the voltage value, in volts, for the operating conditions. Allowed values are 0.0 through 1000.0.

`-tree_type` *tree_type*

Specifies the tree type for the operating conditions. Allowed values are *best_case_tree*, *balanced_tree* (the default), or *worst_case_tree*. The tree type is used to estimate interconnect delays by providing a model of the RC tree.

`-calc_mode` *`calc_mode`*

For use only with DPCM libraries. Specifies the DPCM delay calculator mode for the operating conditions; analogous to the *-process* option used in Synopsys libraries. Allowed values are *unknown* (the default), *best_case*, *nominal*, or *worst_case*. The default behavior (*unknown*) is to use worst case values during analysis similarly to worst_case. If *-rail_voltages* are specified, the command sets all (worst_case, nominal, and best_case) voltage values.

`-rail_voltages` *`rail_value_pairs`*

Specifies a list of name-value pairs that defines the voltage for each specified rail. The name is one of the rail names defined in the library; the value is the voltage to be assigned to that rail. By default, rail voltages are as defined in the library; use this option to override the default voltages for specified rails.

Description

This command creates a new set of operating conditions in the specified library. A technology library contains a fixed set of operating conditions; this command allows you to create new, additional operating conditions.

To see the operating conditions defined for a library, use the *report_lib* command.

To set operating conditions on the current design, use the *set_operating_conditions* command.

Examples

The following example creates a new set of operating conditions called WC_CUSTOM in the library "tech_lib", specifying new values for process, temperature, voltage, and tree type. By default, rail voltages remain as defined in the library.

```
pt_shell> create_operating_conditions -name WC_CUSTOM \\
-library tech_lib -process 1.2 -temperature 30.0 -voltage 2.8 \\
-tree_type worst_case_tree
```

The following example creates a new set of operating conditions called OC3, in the library "IBM_CMOS5S6_SC", specifying new values for process, temperature, voltage, and rail voltages for rails VTT and VDDQ. By default, the tree type *balanced_tree* is used.

```
pt_shell> create_operating_conditions -name OC3 \\
-lib IBM_CMOS5S6_SC -proc 1.0 -temp 100.0 -volt 4.0 \\
-rail_voltages {VTT 3.5 VDDQ 3.5}
```

See Also

- set_operating_conditions
- report_lib

Feedback

create_path_tag_set

Associates a named tag with timing paths in a collection.

Syntax

```
status create_path_tag_set
  [path_collection]
  [-append]
  [-name tag_name]
```

Data Types

```
path_collection     collection
tag_name            string
```

Arguments

path_collection

Specifies a collection of timing paths to tag. If not specified, an empty tag set is created.

`-name` *tag_name*

Specifies the tag name. Without this option, the command uses the name "default".

`-append`

Adds the specified paths to the existing tag set. Without this option, the command overwrites an existing tag set with the same name.

Description

The *create_path_tag_set* command provides a capability to tag collections of timing paths. Tagged timing paths can be excluded from appearing in the output of subsequent calls to *get_timing_paths* or *report_timing*.

You can specify the name of the tag using the *-name* option. The default name is "default".

If *path_collection* option is not specified, the command creates a new empty tag set if there is not already a tag set of the specified name.

Limitations

Collection tagging only applies to collections of timing paths.

Feedback

Examples

The following example tags a collection of timing paths and excludes the tagged timing paths from a subsequent *get_timing_paths* command:

```
pt_shell> set_app_var enable_path_tagging true
true
pt_shell> set mypaths [get_timing_paths ...]
pt_shell> create_path_tag_set $mypaths -name tag1
1
pt_shell> set more_paths [get_timing_paths -nworst 5 -max 5 -exclude
 tag1]
_sel2
```

The following example shows an iterative process where timing paths are analyzed in batches of 1000 and previously analysed paths are excluded from new results. When the *create_path_tag_set* command is used with the name of an existing tag set, the *-append* option causes the paths to be added to the existing tag set.

```
pt_shell> set path_set1 [get_timing_paths -max_paths 1000]
_sel1
pt_shell> report_timing -pba_mode path $path_set1
pt_shell> create_path_tag -name Alrdy_analyzed $path_set1
pt_shell> set path_set2 \\
 [get_timing_paths slack 0 max_paths 1000 -exclude Alrdy_analyzed]
_sel2
pt_shell> report_timing -pba path $path_set2
pt_shell> create_path_tag -append -name Alrdy_analyzed $path_set2
pt_shell> report_path_tag_set
****************************************
Report : Timing Path Tag Sets
 ...
****************************************

Tag name              |  Number of paths
------------------------------------------
Alrdy_analyzed        |  2000

1
```

See Also

- get_timing_paths
- remove_path_tag_set
- report_path_tag_set
- enable_path_tagging

Feedback

create_placement_blockage

Creates a new placement blockage.

Syntax

status *create_placement_blockage*

```
-bbox coordinate_list
[-name blockage_name]
```

Data Types

```
coordinate_list   list
blockage_name     string
```

Arguments

`-bbox coordinate_list`

Specifies the lower-left and upper-right coordinates (in microns) of the bounding box of the placement blockage in the following format: {*lower_left_x lower_left_y upper_right_x upper_right_y*}.

`-name blockage_name`

Specifies the optional name of the blockage.

Description

This command creates a new placement blockage that controls the placement of ECO cells when you use the *fix_eco_drc* or *fix_eco_timing* command with the *-physical_mode* option. PrimeTime does not resize cells or insert buffers within placement blockages.

You can create a placement blockage only after PrimeTime loads the physical database. To specify the file that contains *create_placement_blockage* commands, use the *set_eco_options* command with the *-physical_constraint_file* option. When you use this option, PrimeTime loads the physical database, reads the constraint file, and creates blockage areas before fixing violations.

Examples

The following example creates a placement blockage named placeblockage_10.

```
pt_shell> create_placement_blockage -bbox {0 0 100 100} -name
 placeblockage_10
```

See Also

- create_voltage_area
- fix_eco_drc

Feedback

- fix_eco_timing
- set_eco_options
- write_changes

---

create_power_domain

Creates a power domain at the specified scope, which provides a power supply distribution network.

Syntax

string *create_power_domain*

```
domain_name
[-elements element_list]
[-exclude_elements exclude_list]
[-include_scope]
[-scope instance_name]
[-supply supply_set_handle_or_ref]
[-available_supply supply_set_handle_or_ref]
[-update]
[-atomic]
```

Data Types

domain_name	string
element_list	list
exclude_list	list
instance_name	string
supply_set_handle_or_ref	string

Arguments

domain_name

Specify the name of the power domain to be created. The name should be a simple (non-hierarchical) name. You must specify this argument.

If there is a power domain with the same name in the specified scope, the power domain cannot be created. If there is a hierarchical/leaf cell instance or port with the same name in the specified scope, the power domain cannot be created either.

-elements *element_list*

Specify a list of cells which are added as the extent of the power domain. The list of cells should be hierarchical, macro, or I/O pad cells, but PrimeTime accepts any cell. Specified cells cannot be added in other power domains of the same scope.

If neither *-elements* nor *-include_scope* is specified, the power domain consists of the current scope and any of its children not specified as elements in another *create_power_domain* command.

`-exclude_elements` *`exclude_list`*

Specify a list of design elements to exclude from the effective_element_list.

`-include_scope`

If this option is specified, the scope of the power domain is also included in the extent of the power domain. This means all elements within the current scope get the supply as the power domain.

`-atomic`

Defines the minimum extent of the power domain. PrimeTime currently reads and ignores this option.

`-scope` *`instance_name`*

Specify in which scope the power_domain is going to be created. The instance name is the name of a hierarchical cell.

If this option is not specified, the power domain is created in the current scope.

`-supply` *`supply_set_handle_or_ref`*

Specifies the *supply_set_ref* for the domain. *supply_set_handle_or_ref* follows the following format : {*supply_set_handle* [*supply_set_ref*]}. If *supply_set_ref* is also specified, the domain supply_set_handle is associated with the specified supply_set_ref. This option can be specified multiple times.

`-available_supply` *`supply_set_handle_or_ref`*

PrimeTime currently reads and ignores this option.

Either predefined or user-defined handle names can be used. The predefined supply_set_handles are:

- *primary*: primary supply set of this power domain
- *default_isolation*: default isolation supply set for any isolation strategy applied to this power domain if the isolation power or ground is not specified in the strategy
- *default_retention*: default retention supply set for any retention strategy applied to this power domain if the retention power or ground is not specified in the strategy

Handles of the form *extra_supplies_[0-9]+* are accepted but ignored in PrimeTime.

`-update`

Updates the supply set association of an existing power domain.

The *-update* option is used along with and only with the *domain_name* and *-supply* options. It is an error to use the *-update* option with any other option.

This option allows you to

- Add a new supply set handle name without explicitly defining a supply set association
- Add a supply set to be associated with a previously declared supply set handle
- Add a new supply set handle and a supply set association in a single command
- Add cells to a previously created domain.

NOTE: It is an error to update a supply set handle that you have already associated with a supply set. It is an error to add cells that are already root cells of the domain.

Description

The *create_power_domain* command enables you to create a power domain in the specified scope. A power_domain is a collection of design elements that share a primary power and ground power net. The logic hierarchy level where a power domain is created is called the scope of the power domain. Any design elements that belong to a power domain are said to be in the extent of that power domain. While a design element can be in the scope of a number of power domains, if can only be in the extent of one power domain.

A power domain could have several supply_nets, supply_nets are connected to power_domain via supply_port. And a power_switch of a power_domain could be used to turn on/off the power supply of part/whole power domain. The supply net, supply port, and power switch can be created using the *create_supply_net*, *create_supply_port* and *create_power_switch* commands respectively.

This command returns collection object (the newly created power domain). It returns null string if it failed.

Non-UPF Mode (Power Domains Mode): This command works in power domains mode as well, and take following options instead.

string *create_power_domain*

```
domain_name
[-power_down]
[-power_down_ctrl list]
```

```
[-power_down_ack list]
[-power_down_ctrl_sense <0 or 1>]
[-object_list list]
```

-power_down

Use this to specify the new power domain is a power down domain. This option is a prerequisite for the *-power_down_ctrl* option.

-power_down_ctrl *list*

Use this to specify the power down control net for the new power domain. The *-power_down* option is required in order to use this option. It is also a prerequisite for the *-power_down_ack* option.

-power_down_ack *list*

Use this to specify the power down acknowledge net for the new power domain. The *-power_down_ctrl* option is required in order to use this option.

-power_down_ctrl_sense *<0 or 1>*

Use this to specify the sense of the power down control net for the new power domain. It take values of 0 or 1. It specifies the active low or active high sense type for power control signal. The *-power_down_ctrl* option is required in order to use this option.

-object_list *list*

A list of hierarchical cells that are associated with the new power domain. When this option is absent, the new power domain is assumed to be the top level domain. For each design, there can only be one top-level domain. Also, each hierarchical cell can only be associated with one power domain.

The *create_power_domain* command creates a new power domain for the current design. The power domains for each design must be uniquely named. There can be only one design-wise top level power domain.

Use *-power_down* to specify a new domain is a power down domain. An always-on power domain is created without this switch. For a power down domain, use *-power_down_ctrl* to specify an optional power down control net; use *-power_down_ack* to specify an optional power down acknowledge net.

Use *-object_list* to associate a list of hierarchical cells with the new power domain. When this option is not used, the new power domain is considered as top-level power domain. Each hierarchical cell can only belong to one power domain.

By default, power down control and power acknowledge signals are marked dont touch. Set the *dont_touch_power_domain_control_nets* variable to false if you do not want these signal marked dont_touch.

Feedback

Examples

UPF Mode:

The following example creates two power_domains in scope INST1:

```
pt_shell> create_power_domain PD1 -elements INST1/SUB_INST -scope INST1
{"INST1/PD1"}

pt_shell> create_power_domain PD2 -elements INST1/SUB_INST -scope INST1
Error: Cell 'INST1/SUB_INST' is already in the extent of power domain
 'PD1'. (UPF-001)

pt_shell> create_power_domain PD2 -scope INST1 -include_scope
{"INST1/PD2"}
```

The following example excludes instance INST1 specified in -exclude_elements option from the effective element list:

```
pt_shell> create_power_domain PD_TOP -elements {INST1 INST2}
 -exclude_elements {INST1}
```

The following example creates a power domain in scope INST1, and uses a supply set to provide the default primary power and ground, and a different supply set to provide the default retention power and ground nets.

```
pt_shell> create_power_domain PD3 -scope INST1 -supply {primary set1} \\
            -supply {default_retention set2}
```

The following example updates an existing power domain with the *-supply* option.

```
prompt> create_power_domain PD_MID -scope mid1 -supply {abc}
mid1/PD_MID
prompt> create_power_domain PD_MID -scope mid1 -update \\
    -supply {primary SS1} -supply {abc SS1}
mid1/PD_MID
```

Non-UPF Mode:

The following examples use the command to create power domains.

```
pt_shell> create_power_domain TOP_DOMAIN
1

pt_shell> create_power_domain SUB_DOMAIN -power_down \\
                  -power_down_ctrl [get_nets pd_ctrl] \\
                  -power_down_ack [get_nets pd_ack] \\
                  -object_list [get_cells mid1]
1
```

Feedback

See Also

- report_power_domain

---

create_power_group

Creates a power group of cells in the current design.

Syntax

int *create_power_group*

```
-name group_name
-default
object_list
```

Data Types

group_name	string
object_list	list

Arguments

`-name` *group_name*

Specifies the name for the power group. The name should not conflict with names of existing power groups.

`-default`

Indicates that the specified power group is predefined and the default list of cells are to be included in the power group. This option is mutual exclusive with the *object_list* option.

object_list

Specifies a list of cells to be included in this power group. This option is mutual exclusive with the *-default* option.

Description

Group a list of cells of special interests (not necessarily in the same hierarchy) to form a power group. The *update_power* and *report_power* commands are able to generate power waveform and power report for this power group.

The cells contained in power groups are leaf cells. If a hierarchical cell is specified in the *object_list* option, all the leaf cells contained by the hierarchical cell, instead of the hierarchical cell itself, are added to the power group.

There are several power groups predefined and automatically created by the tool. However, if a predefined power group has been removed, the *-default* option can be used to regenerate it.

The following is a list of the predefined power groups ordered by priority from high to low. Note that the predefined power groups are not supposed to be overlapping with one another. If one cell happens to belong to more than one group, it is put into the group with the higher priority.

- *io_pad* - All I/O PAD cells.
- *memory* - All memory cells.
- *black_box* - All black boxes.
- *clock_network* - Cells in the clock network. The contents is consistent with the results from the *get_clock_network_objects -type cell* command. If the *power_clock_network_include_clock_gating_network* variable is set to *true*, the contents is consistent with the results from the *get_clock_network_objects -type cell -include_clock_gating_network* command.
- *register* - The latches, flip-flops driven by the clock network. The contents is consistent with the results from the *get_clock_network_objects -type register* command. If the *power_clock_network_include_clock_gating_network* variable is set to *true*, the contents is consistent with the results from the *get_clock_network_objects -type register -include_clock_gating_network* command.
- *sequential* - All other sequential cells.
- *combinational* - All other combinational cells.

To move cells from one predefined group to another, add the power_cell_type attribute and assign the value to the desired predefined group. The power_cell_type attribute takes precedence over the rules used to determine which group a cell belongs to.

Examples

In the following example, a power group called clock_tree is created to accommodate all the clock tree cells.

```
pt_shell> create_power_group -name clock_tree [get_clock_network_objects
 -type cell]
1
```

In the following example, the predefined power group clock_network has been removed earlier, now you want to create it again with the default list of cells.

```
pt_shell> create_power_group -name clock_network -default
1
```

Feedback

See Also

- report_power_groups
- remove_power_groups
- get_power_group_objects
- update_power
- report_power
- power_clock_network_include_clock_gating_network

---

create_power_net_info

Creates a power net.

Syntax

int *create_power_net_info*

```
power_net_name
-power
-gnd
[-switchable]
[-nominal_voltages nominal_voltage_list]
[-voltage_ranges voltage_range_list]
```

Data Types

```
power_net_name          string
nominal_voltage_list    list
voltage_range_list      list
```

Arguments

power_net_name

The name of the new power net info to be created.

-power

Specifies the type of the new power net is "power". Mutually exclusive with -gnd option.

-gnd

Specifies the type of the new power net is "gnd". Mutually exclusive with -power option.

Feedback

`-switchable`

Specifies that this power net is switchable (can be cut off externally, or, if driven by an internal switch, cut off internally). This option can only be used with -power.

`-nominal_voltages` *`nominal_voltage_list`*

Specifies the list of nominal voltages this power net has been designed to operate. The actual operating voltage of the power net can deviate from the nominal within a certain tolerance as specified by *voltage_ranges_list*. This optional argument must be specified together with *voltage_ranges_list*. This option can only be used with -power.

`-voltage_ranges` *`voltage_range_list`*

Specifies the list of allowed voltage ranges around the nominal voltages that this power net has been designed to operate. This optional argument must be specified together with *nominal_voltages_list*. This option can only be used with -power.

Description

The *create_power_net_info* command creates a new power net info for the current design. The power net is either of type power or of type gnd.

Examples

The following examples use the command to create power nets.

```
prompt> create_power_net_info VSS_net -gnd
1
prompt> create_power_net_info VDD1_net -power
1
prompt> create_power_net_info VDD2_net -power \\
   -switchable -nominal_voltages {0.9 1.08} \\
   -voltage_ranges {0.88 0.92 1.05 1.10}
1
```

See Also

- report_power_net_info
- set_voltage

---

create_power_rail_mapping

Maps the power rails defined in the libraries to the physical power rails existing in the design.

Feedback

Syntax

status create_power_rail_mapping

```
[-lib_rail_name power_rail_name]
[-cells cell_list]
[-off_condition condition]
[-default]
design_rail
```

Data Types

power_rail_name	string
cell_list	list
condition	string
design_rail	string

Arguments

`-lib_rail_name` *power_rail_name*

Specifies the power rail defined in the library. It can take the power rail name defined in the library and also take "all". The power rail name can be the name of the *voltage_map* command defined in the library. By using the *-lib_rail_name all* option, the mapping is applied to all the power rails in each library associated with the library cells. If the *-lib_rail_name* option is not used, the mapping is applied to the default library power rail in each library associated with the library cells.

`-cells` *cell_list*

Specifies the instance names or collections of instances. The instance can be hierarchical or leaf instances. For an instance block, only the top instance needs to be specified. All the instances inside such block will be affected by the mapping. Without this option, the mapping will be created for the whole design.

`-off_condition` *condition*

Specifies the power-off condition for the design power rail. At power-off state, the portion of design that is powered off by the specific design power rail will not contribute any dynamic or static power. The Boolean expression takes net names plus operators. The operators supported are: (,), !, +, *, &, |, ^, '. When the expression is evaluated to be "1", the design power rail is in power-off state. If the expression is evaluated to be "0" or any unknown state, the design power rail is in power-on state. Ensure that the states of the nets used in the Boolean expression are clearly defined to avoid any unexpected results. If the *-power_off* option is omitted, the design power rail is assumed to be on all the time.

When switching activity information is specified by the VCD file, the state of the power control signal nets are monitored. The power-off expression is evaluated if there is any state change on these control nets. When a specific power rail is

powered-off, neither dynamic nor static power will be dissipated on the cell (or part of cell) powered by this rail.

When switching activity information is specified by statistical toggle rate and state probability, the state probability of the power rail being on is calculated. As a default, only leakage power is scaled by the state probability of the associated power rail being on. Set the *power_scale_dynamic_power_at_power_off* variable to true, if dynamic power needs to be scaled too. Dynamic power scaling is only needed if the statistical switching activity includes toggles happened when the rail is being powered off.

`-default`

Specifies the design power rail to be the default rail in the design. If not specified, the design power rail defined by the first *create_power_rail_mapping* command is assigned as the default design rail.

`design_rail`

Defines the power rail used in the design.

Description

Use this command to map the power rails defined in the libraries to the physical power rails existing in the design. Such rail mapping is done at cell (instance) level, which gives you the flexibility to connect the same library cell to different physical power rails in a design. The design power rails specified in this command are virtual power rails, which means that the rail connections do not actually exist in the netlist. No voltage level violation will be checked for such virtual rail connections. It is allowed to connect cells with different voltages to the same virtual design power rail. This command enables PrimePower to create collections of instances on different power rails. Thus, provides the capability and flexibility to create rail-based power reporting at design level. This command also enables PrimePower to be sensitive to the power-on/off control signals associated with the design power rails.

Single-rail or multi-rail cell in PrimePower is based on its cell definition in logic library. A multi-rail cell can have rail specific internal or leakage power tables defined in the library. So that power consumption can be reported for different rails.

This command can create the rail mapping for the whole design or for specific instances (hierarchical or leaf-level). Multiple *create_power_rail_mapping* commands can be issued, the latter overwrites the previous settings if it overlaps.

For single-rail design, if not specified, the default library rail will be mapped to the default design rail internally.

The *report_power_rail_mapping* command can be used to report existing power rail mapping information.

Feedback

Examples

The following example shows how to create rail mapping and generate rail-based power reports for a multi-rail design. There are three instance blocks in this design: block1, block2 and ls_block. Block block1 and block2 are both single-rail instance blocks with all the leaf cells powered by the default library power rail defined in each associated logic library. Block block1 is connected to design power rail VDD1 in the design while block block2 is connected to design power rail VDD2 in the design. Block ls_block is a dual-rail instance block with rail V1 and V2 defined in its cells' logic library. Library power rail V1 is connected to design power rail VDD1 and library power rail V2 is connected to design power rail VDD2. The *current_power_rail* command can select the rail of interest and the *update_power* command can generate the power analysis results just for the interested rails. As a result, report Power_VDD1.rpt file will only contain the power consumed on design power rail VDD1 and report Power_VDD2.rpt will only contain the power consumed on design power rail VDD2. Since the *current_power_rail all* command will mark all the design power rails to be interested, report Power_all.rpt file will contain the total power consumed for entire design.

```
pt_shell> create_power_rail_mapping VDD1 -cells block1
pt_shell> create_power_rail_mapping VDD2 -cells block2
pt_shell> create_power_rail_mapping VDD1 -lib_rail_name V1 -cells
 ls_block
pt_shell> create_power_rail_mapping VDD2 -lib_rail_name V2 -cells
 ls_block
pt_shell> current_power_rail VDD1
pt_shell> update_power
pt_shell> report_power > Power_VDD1
pt_shell> current_power_rail VDD2
pt_shell> update_power
pt_shell> report_power > Power_VDD2
pt_shell> current_power_rail all
pt_shell> update_power
pt_shell> report_power > Power_all
```

See Also

- current_power_rail
- read_saif
- read_vcd
- report_power
- report_power_rail_mapping
- set_switching_activity

Feedback

- update_power
- power_scale_dynamic_power_at_power_off

---

create_power_state_group

Defines a group name to be used in the *add_power_state* command.

Syntax

status *create_power_state_group*

```
        group_name
```

Data Types

group_name	string

Arguments

group_name

Specifies the name of the group to create. It must use a simple name.

Description

The *create_power_state_group* command defines a group in the active scope that can be used with the *-group* option of the *add_power_state command*.

Use a group to collect related power states defined by the *add_power_state* command. The legal power states of a group define the legal combinations of power states of other objects in this scope or the descendant subtree, that is, those combinations of states of those objects that can exist at the same time during operation of the design.

It is an error if the specified group name already exists in the active scope.

Examples

The following example shows how to define a power state group.

```
prompt> create_power_state_group PST_GROUP
```

See Also

- add_power_state

Feedback

create_power_switch

Creates a power switch at the specified power domain. This command is supported only in the UPF mode.

Syntax

status *create_power_switch*

```
switch_name
-domain domain_name
-output_supply_port {port_name supply_net_name}
-input_supply_port  {port_name supply_net_name}
-control_port {port_name net_name}
-on_state {state_name input_supply_port {boolean_function}}
[-off_state {state_name {boolean_function}}]
[-on_partial_state {state_name input_supply_port {boolean_function}}]
[-error_state {state_name {boolean_function}}]
[-ack_port {port_name net_name [{boolean_function}]}]
[-ack_delay {port_name delay}]
[-supply {port_name supply_net_name}]
```

Data Types

```
switch_name           string
domain_name           string
port_name             string
supply_net_name       string
net_name              string
state_name            string
input_supply_port     string
boolean_function      list
delay                 string
supply_net_name                string
```

Arguments

switch_name

Specifies the power switch to create. The name should be a simple (nonhierarchical) name. You cannot create a power switch with the same name as the existing power switch in a specified power domain. This option is required.

-domain *domain_name*

Specifies the power domain that contains the power switch. If the power domain with the specified name does not exist in the current scope, the command fails. This option is required.

`-output_supply_port {port_name supply_net_name}`

Specifies the output port of the power switch and the supply net where the port connects. The command fails when,

- A port exists with the same name on the power switch.
- A supply net with the same name does not exist in the current scope. This option is required.

`-input_supply_port {port_name supply_net_name}`

Specifies the input port of the power switch and the supply net where the port connects. The command fails when,

- A port exists with the same name on the power switch.
- A supply net with the same name does not exist in the current scope. This option is required.

`-control_port {port_name net_name}`

Specifies the control port of the power switch and the logical net where this port connects. The command fails when,

- A port with the specified name already exists on the power switch.
- A net with the specified name does not exist. This option is required. You can specify this option more than one time. One power switch can have multiple control ports.

`-on_state {state_name input_supply_port {boolean_function}}`

Specifies the on state, the relevant input supply port, and its Boolean function. You can specify this option multiple times.

`-off_state {state_name {boolean_function}}`

Specifies the off state and its relevant Boolean function. This option is currently ignored by the PrimeTime tool.

`-on_partial_state {state_name input_supply_port {boolean_function}}`

Specifies the partial state and its relevant Boolean function. This option is currently ignored in the PrimeTime tool.

`-error_state {state_name {boolean_function}}`

Specifies the error state and its relevant Boolean function. This option is currently ignored by the PrimeTime tool.

`-ack_port {port_name net_name [{boolean_function}]}`

Specifies the acknowledge port of the power switch and the logical net where this port connects. The command fails when,

- A port with the specified name already exists on the power switch.
- A net with the specified name does not exist. You can specify a Boolean function. This option is currently ignored by the PrimeTime tool.

`-ack_delay {port_name delay}`

Specifies the acknowledge port on the switch and the corresponding delay. This option is currently ignored by the PrimeTime tool.

`-supply {port_name supply_net_name}`

PT currently reads and ignores this option.

Description

This command enables you to create a power switch at the specified power domain. The switch is created within the scope of the power domain. Each power switch must be connected with an input supply net and an output supply net. The power switch can be connected with an acknowledge net and several control nets through a switch port. The switch ports are created if the power switch is created successfully, otherwise generates an appropriate error message.

Examples

The following example creates power switch, SW1 within power domain, PD1:

```
pt_shell> create_power_switch SW1 -domain PD1 \\
          -output_supply_port {vout VNO2} \\
          -input_supply_port {vin1 VNI1} \\
          -control_port {ctrl_small ON1}  \\
          -on_state {full_s vin1 {ctr_small}}
```

The following example generates an error message, when the command fails.

```
pt_shell> create_power_switch sw1 -domain ADD1 \\
          -control_port {ctrl power_down} \\
          -input_supply_port {in VDD1} \\
          -output_supply_port {out VDD1_ADD} \\
          -on_state {state_2001 in {!ctrl}}
Warning: Nothing implicitly matched 'ADD1' (SEL-003)
Error: Nothing matched for collection (SEL-005)
Error: The '-domain' option of create_power_switch command must specify
exactly one power domain object. (UPF-004)
0
```

Feedback

See Also

- get_power_switches
- report_power_switch

---

create_pst

Creates a power state table (PST), using a specific order of supply nets.

Syntax

string *create_pst*

```
table_name
-supplies list
```

Data Types

```
table_name          string
list                list
```

Arguments

table_name

Specifies the name of the power state table.

This argument is required.

-supplies *list*

Specifies a list of supply nets to include in each power state table in the design.

This option is required.

Description

The *create_pst* command creates a power state table using a specific order of supply nets. A power state table is used for implementation specifically for synthesis, analysis, and optimization. The power state table defines the legal combinations of states, which are those combinations of states that can exist at the same time during the operation of the design.

An error occurs if a specified supply net has not been created before the *create_pst* command is run.

Examples

The following is an example of the *create_pst* command:

```
prompt> create_pst MyPowerStateTable -supplies {PN1 PN2 SOC/OTC/PN3}
```

Feedback

See Also

- add_power_state
- add_pst_state

---

create_qtm_constraint_arc

Creates a constraint arc for a quick timing model.

Syntax

string *create_qtm_constraint_arc*

```
[-name arc_name]
[-setup]
[-hold]
[-recovery]
[-removal]
-from port_name
[-to port_spec]
-edge rise | fall
[-signal_edge rise | fall]
[-path_type name]
[-path_factor multiplication_factor]
[-value constraint_value]
```

Data Types

arc_name	string
port_name	string
port_spec	list
name	string
multiplication_factor	float
constraint_value	float

Arguments

`-name arc_name`

Specifies the name of the constraint arc.

`-setup`

Creates a setup arc.

`-hold`

Creates a hold arc.

`-recovery`

Creates a recovery arc.

`-removal`

Creates a removal arc.

`-from` *`port_name`*

Specifies the constraining port name. Define this port as a clock port.

`-to` *`port_spec`*

Specifies the constrained port. Define this port as an input/inout port.

`-edge rise | fall`

Specifies the triggering edge of the clock (rise or fall).

`-signal_edge rise | fall`

Specifies the direction of the signal at the constrained port specified by the *-to* option.

If you do not specify this option, the tool creates constrinats for both signal rise and signal fall.

`-path_type` *`name`*

Specifies the path type that you want to use to set the delay.

`-path_factor` *`multiplication_factor`*

Specifies the multiplication factor for the path type.

`-value` *`constraint_value`*

Specifies the delay value in terms of the time units you use for the model.

Description

This command allows you to create a constraint arc in a quick timing model. The *-from* port denotes the constraining port, and the *-to* port denotes the constrained port. The *-from* port must be defined as a clock port and the *-to* port must be an input or inout port. You can use *-signal_edge* option to specify the the direction of the signal at the constrained port specified by the *-to* option.

Use the *-setup*, *-hold*, *-recovery* and *-removal* options to specify a setup, hold, recovery or removal arc. You must use one of the switches, but you cannot use them simultaneously. To specify the triggering edge of the clock, use the *-edge* option and specify *rise* or *fall*.

To specify the delay value, use the *-path_type* option. You can specify the delay as a value in terms of the time units used in the design. Use either the *-path_type* or the *-value* option; you cannot use both simultaneously.

If the constraint arc is a setup/hold/recovery/removal arc, the quick timing model (QTM) parameter global setup/hold/recovery/removal time is added to the value of the specified

Feedback

type correspondingly. You must define the global setup/hold/recovery/removal time before creating any arc of the correspongin type. The command checks to see if these parameters are defined. Global setup/hold/recovery/removal time are defined using the *set_qtm_global_parameter* command.

To see the information about the current quick timing model, use the *report_qtm_model* command.

For a basic description about quick timing models, see the *create_qtm_model* man page.

Examples

The following command creates a setup arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay equivalent to 2 times that of path type path1.

```
pt_shell> create_qtm_constraint_arc -setup -edge rise -from CLK -to IN1
            -path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay equivalent to 2 times that of path type path1.

```
pt_shell> create_qtm_constraint_arc -hold -edge rise -from CLK -to IN1
            -path_type path1 -path_factor 2
```

The following command creates a hold arc from the positive edge of constraining pin CLK to IN1 (constrained pin) with a delay of 3 time units.

```
pt_shell> create_qtm_constraint_arc -hold -edge rise -from CLK -to IN1
 -value 3.0
```

See Also

- create_qtm_delay_arc
- create_qtm_model
- create_qtm_path_type
- remove_qtm_constraint_arc
- get_qtm_ports
- report_qtm_model
- save_qtm_model

---

create_qtm_delay_arc

Creates a delay arc for a quick timing model (QTM).

Feedback

Syntax

string *create_qtm_delay_arc*

```
[-name arc_name]
-from port_spec
-to port_spec
[-edge rise | fall]
[-from_edge rise | fall]
[-to_edge rise | fall]
[-path_type path_type]
[-path_factor multiplication_factor]
[-value delay_value]
[-total_value total_value]
```

Data Types

arc_name	string
port_spec	list
path_type	string
multiplication_factor	float
delay_value	float
total_value	float

Arguments

`-name` *arc_name*

Specifies the name of the delay arc.

`-from` *port_spec*

Specifies the QTM port name or a collection of QTM ports, which is the startpoint of the delay arc. For an edge triggering arc this must be a clock.

`-to` *port_spec*

Specifies the QTM port name or a collection of QTM ports. This port must be output or inout type.

`-edge rise | fall`

Specifies the triggering edge (rise or fall)

`-from_edge rise | fall`

Specifies the triggering edge (rise or fall) for clock start points or the signal direction for data startpoints. If you do not use this option, the command creates delay arcs from both rise and fall edges at the start points.

`-to_edge rise | fall`

Specifies the signal direction at the data endpoints. If you do not use this option, the command creates delay arcs to both rise and fall edges at the endpoints.

If you omit both the *-from_edge* and *-to_edge* options, the command creates *rise-to-rise* and *fall-to-fall* delay arcs.

`-path_type` *`path_type`*

Specifies the path type you want to use to set the delay.

`-path_factor` *`multiplication_factor`*

Specifies the multiplication factor for the path type.

`-value` *`delay_value`*

Specifies the delay value in terms of the time units you use for the model.

`-total_value` *`total_value`*

Specifies the total delay value in terms of the time units used for the design. The drive arc delay and load are not considered when determining the total delay value for the arc.

Description

This command allows you to create a delay arc in a QTM from a QTM port or collection of QTM ports to a port or collection of QTM ports. If you specify a list of ports for the *-from* and *-to* ports, a delay arc is created from each start port to each end port.

To create an edge triggering arc, use the *-edge* option. For an edge triggering arc, the *clk_to_output* delay (specified as a global parameter) is added to the delay value specified. You can use *-from_edge* and *-to_edge* to specify the triggering edge for clock start points or the signal direction for data startpoints and endpoints.

You can use the *-path_type* option to specify the delay value or you can specify the delay as a value in terms of the time units used in the design. Use either the *-path_type* or the *-value* option; you cannot use both options simultaneously.

To show the information about the current QTM model, use the *report_qtm_model* command.

For a basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following command creates a delay arc from input ports {A, B, C} to output port D with a delay value of 2.0 time units.

```
pt_shell> create_qtm_delay_arc -from {A, B, C} -to D -value 2.0
```

The following command creates an arc from clock CLK to output port OUT with a delay equivalent to 3 times that of path type 'path1'.

```
pt_shell> create_qtm_delay_arc -from CLK -to OUT -path_type path1
 -path_factor 3
```

Feedback

The following example uses wildcard to match all ports matching "CL*".

```
pt_shell> create_qtm_delay_arc -from CL* -to OUT -path_type path1
 -path_factor 3
```

The following example uses a collection for the QTM port 'CLK':

```
pt_shell> create_qtm_delay_arc -from [get_qtm_ports CLK] -to OUT
          -path_type path1 -path_factor 3
```

See Also

- create_qtm_constraint_arc
- create_qtm_model
- create_qtm_path_type
- remove_qtm_delay_arc
- get_qtm_ports
- report_qtm_model
- save_qtm_model

---

create_qtm_drive_type

Creates a drive type in a quick timing model (QTM) description.

Syntax

string *create_qtm_drive_type*

```
-lib_cell lib_cell_name
[-input_pin input_pin_name]
[-output_pin output_pin_name]
[-input_transition_rise rtrans]
[-input_transition_fall ftrans]
drive_type_name
```

Data Types

```
lib_cell_name    string
input_pin_name   string
output_pin_name  string
rtrans           float
ftrans           float
drive_type_name  string
```

Feedback

Arguments

`-lib_cell` *`lib_cell_name`*

Specifies the library cell name in the technology library.

`-input_pin` *`input_pin_name`*

Specifies the input pin in the *lib_cell_name* is the startpoint of the arc and ends in the output pin.

`-output_pin` *`output_pin_name`*

Specifies the output pin in the *lib_cell_name* that specifies the drive.

`-input_transition_rise` *`rtrans`*

Specifies the input rising transition time associated with the *-input_pin* option to compute the delay for the drive arc. If you do not include this option, the delay table for rising input of the drive arc are loaded from library as is.

Use the *-input_transition_rise* and *-input_transition_fall* options to capture the transition time associated with the *input_pin* option. This can obtain more accurate information on the transition time and delay time for the defined drive arc.

`-input_transition_fall` *`ftrans`*

Specifies the input falling transition time associated with the *-input_pin* option to compute the delay for the drive arc. If you do not include this option, the delay table for falling input of the drive arc are loaded from library as is.

`drive_type_name`

Specifies the name given to the defined drive type.

Description

This command can be used to create a drive type. The defined drive type can be used to set the drives on output ports. The drive type is specified by referring to a lib_cell in the technology library. The library cell is specified by using the *-lib_cell* option. This library cell must be present in the technology library.

The output pin, which drives the port, is specified by using the *-output_pin* option. In addition, you can specify the input pin by using the *-input_pin* option. This option selects an arc that drives the output pin of the library cell.

If neither the input pin nor the output pin is specified, an arbitrary arc is chosen to define the drive type. If the input pin is specified and output pin is not specified, an arbitrary arc starting from the input pin defines the drive type. If the output pin is specified and input pin is not specified, the arbitrary delay arc coming into the output pin defines the drive type.

If both input and output pins are specified, the library cell must have an combinational arc from the input pin to the output pin.

To use this command, you must read in the technology library and then specify it by using the *set_qtm_technology* command.

To show the information about the current quick timing model, use the *report_qtm_model* command.

For a basic description quick timing models, see the *create_qtm_model* man page.

Examples

In the following examples it is assumed that you have loaded the technology library, and the quick timing model technology is set. Do this by using the following the sequence of commands as a guide, substituting the name of your technology library.

```
pt_shell>read_db my_technology_library.db
```

```
pt_shell>set_qtm_technology -library my_technology_library
```

The following command creates a drive type equivalent to the BUF1 library cell.

```
pt_shell> create_qtm_drive_type -lib_cell BUF1 drive1
```

The following command creates a drive2 drive type equivalent to BUF2 library cell and specifies the output pin to use.

```
pt_shell> create_qtm_drive_type -lib_cell BUF2 -output_pin Y drive2
```

See Also

- create_qtm_load_type
- create_qtm_model
- create_qtm_path_type
- report_qtm_model
- save_qtm_model
- set_qtm_port_drive
- set_qtm_technology

---

create_qtm_generated_clock

Creates a QTM generated clock.

Feedback

Syntax

string *create_qtm_generated_clock*

```
-source master_clock_name
[-divide_by divide_factor | -multiply_by multiply_factor]
[-invert]
generated_clock_name
```

Data Types

```
master_clock_name       string
divide_factor           int
multiply_factor         int
generated_clock_name    string
```

Arguments

`-source master_clock_name`

Specifies the name of the clock defined as master source of the generated clock. The master source must be defined as a clock, or a generated clock, prior to the generated clock definition.

`-divide_by divide_factor`

Specifies the frequency division factor. If the *divide_factor* value is 2, the generated clock period is twice as long as the master clock period.

`-multiply_by multiply_factor`

Specifies the frequency multiplication factor. If the *multiply_factor* value is *3*, the generated clock period is one-third as long as the master clock period.

`-invert`

Inverts the generated clock signal (in the case of frequency multiplication and division).

`generated_clock_name`

Specifies the name of the generated clock. If the name has been used in defining a port or internal pin, then the generated clock is defined on the existing port or internal pin. Otherwise, a new same-named internal pin is created to be the source of the generated clock.

Description

This command creates a QTM generated clock. A generated clock can be defined on the external port of the QTM, as well as the internal pin in the QTM. After defining the generated clock, the source port/pin can be used in other QTM commands the same way as those port/pins defined by the *create_qtm_port* command, i.e. delay arcs from or to the generated clock source pin can be defined with the *create_qtm_delay_arc*

command; constraint arcs related to the generated clock can be defined with the *create_qtm_constraint_arc* command.

For a basic description about QTM, see the *create_qtm_model* man page.

Examples

The following example creates a divide-by 2 generated clock on an internal pin, assume there is no port defined with name "GCLK_int", with master clock CLK.

```
pt_shell> create_qtm_generated_clock GCLK_int -source CLK -multiply_by 2
```

The following example creates a generated clock on port GCLK, assume GCLK has already been defined as a port with command create_qtm_port.

```
pt_shell> create_qtm_generated_clock GCLK -source CLK -divide_by 2
 -invert
```

See Also

- create_qtm_model
- remove_qtm_generated_clock
- get_qtm_ports
- report_qtm_model
- save_qtm_model

---

create_qtm_load_type

Creates a load type for a quick timing model (QTM) description.

Syntax

string *create_qtm_load_type*

```
-lib_cell lib_cell_name
[-input_pin input_pin_name]
load_type_name
```

Data Types

```
lib_cell_name       string
input_pin_name      string
load_type_name      string
```

Feedback

Arguments

`-lib_cell` *`lib_cell_name`*

Specifies the name of the library cell in the technology library.

`-input_pin` *`input_pin_name`*

Specifies the input pin in the lib_cell to specify capacitance.

`load_type_name`

Specifies the name given to the defined load type.

Description

This command is used to create a load type. A load type can be used to define the load on a QTM input port. The load type is specified by referring to the *lib_cell* option in the technology library. The library cell is specified using the *-lib_cell* option. The library cell must be present in the specified technology library.

The input pin can be specified by using the *-input_pin* option. If the input pin is not specified, an arbitrary input pin is chosen to define the load type.

To use this command you must first read in the technology library, then specify the technology library by using the *set_qtm_technology* command.

To show the information about the current quick timing model, use the *report_qtm_model* command.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

In the following example, the technology library is loaded, and the QTM technology is set:

```
pt_shell> read_db my_technology_library.db
pt_shell> set_qtm_technology -library my_technology_library
```

The following command creates a load type load1 using cell OR1. Since the input pin name is not specified, QTM selects an arbitrary input pin.

```
pt_shell> create_qtm_load_type -lib_cell OR1 load1
```

The following command creates a load type load2 using cell OR1, input pin A.

```
pt_shell> create_qtm_load_type -lib_cell OR1 -input_pin A load2
```

See Also

- create_qtm_drive_type
- create_qtm_model

Feedback

- create_qtm_path_type
- report_qtm_model
- save_qtm_model
- set_qtm_port_load

---

create_qtm_model

Begins the definition of a quick timing model (QTM) description.

Syntax

string *create_qtm_model*

```
        model_name
```

Data Types

```
model_name    string
```

Arguments

model_name

Specifies the name of the generated model.

Description

Specifies the name of a new quick timing model (QTM) to be created by PrimeTime. QTMs are temporary timing models that can be created quickly for a block using PrimeTime commands. The purpose of these commands is to provide timing information without the necessity of writing a detailed timing model.

QTMs are intended to be used early in the design cycle to describe rough initial timing of a block. These models will eventually be replaced, either by some other detailed timing model or by the netlist of the block, to obtain more accurate timing.

A QTM can be saved as a Synopsys DB file, which can be instantiated in a design in the same way library cells or ITS models are instantiated. Both Design Compiler and PrimeTime accept designs with instantiated QTMs. To save a QTM model, use the *save_qtm_model* command.

Other commands in the QTM command set must be placed between a *create_qtm_model* and *save_qtm_model* command. Many QTM commands exist for specifying, configuring

Feedback

and examining QTMs. The following is not an exhaustive list, but gives examples of tasks you can perform using QTM commands:

- To create a QTM port, use the *create_qtm_port* command.
- To create constraint arcs and delay arcs, use the *create_qtm_constraint_arc* and the *create_qtm_delay_arc* commands, respectively.
- To set the drive of a QTM output port, use the *set_qtm_port_drive* command.
- To set the load of a QTM input port, use *set_qtm_port_load* command.
- To show the information about the current QTM model, use the *report_qtm_model* command.

Examples

The following command creates a new QTM model with the name adder.

```
pt_shell> create_qtm_model adder
```

See Also

- create_qtm_constraint_arc
- create_qtm_delay_arc
- create_qtm_drive_type
- create_qtm_load_type
- create_qtm_path_type
- create_qtm_port
- get_qtm_ports
- report_qtm_model
- save_qtm_model
- set_qtm_global_parameter
- set_qtm_port_drive
- set_qtm_port_load
- set_qtm_technology

---

create_qtm_path_type

Creates a path type in a quick timing model (QTM) description.

Feedback

Syntax

string *create_qtm_path_type*

```
-lib_cell name
[-input_pin pin_name]
[-output_pin pin_name]
[-fanout count]
path_type_name
```

Data Types

name	string
pin_name	string
count	integer
path_type_name	string

Arguments

`-lib_cell` *name*

Specifies the name of the library cell in the technology library.

`-input_pin` *pin_name*

Specifies the input pin in the *-lib_cell* option, which is the startpoint of the arc, to define the path type.

`-output_pin` *pin_name*

Specifies the output pin in the *-lib_cell* option, which is the endpoint of the arc, to define the path type.

`-fanout` *count*

Specifies the average fanout (number of pins) to consider while computing the delay of the path type. The fanout count can range from 1 to 1000. The default is 1.

path_type_name

Specifies the name given to the defined path type.

Description

This command is used to create a path type. A path type mimics an arc in a library cell. An arc in the QTM model is defined in terms of the delays of the path type. The path type is specified by referring to a library cell in the technology library. The library cell is specified using the *-lib_cell* option. The library cell must be present in the technology library specified.

You can specify the average fanout count the arc fans out to while defining the path type. It is assumed the arc fans out to the first input pin of the same library cell, while calculating

the delays. The input pin is specified by using the *-input_pin* option. The output pin is specified by using the *-output_pin* option.

If neither input pin nor the output pin is specified, an arbitrary delay arc from the library cell is chosen to define the path type.

If the input pin is specified and output pin is not specified, an arbitrary delay arc staring from the input pin defines the path type.

If the output pin is specified and input pin is not specified, an arbitrary delay arc coming into the output pin defines the path type.

If both input and output pins are specified, the library cell must have an arc from the input pin to the output pin.

To use this command you must read in the technology library then specify the technology library by using the *set_qtm_technology* command.

To show the information about the current quick timing model, use the *report_qtm_model* command.

For a basic information about quick timing models, see the *create_qtm_model* man page.

Examples

In the following examples the technology library is loaded by the user and the QTM technology is set. This is done by the following sequence of commands:

```
pt_shell> read_db my_technology_library.db

pt_shell> set_qtm_technology -library my_technology_library
```

The following command creates a path type path1 by using lib_cell AN2, with an average fanout count of 2 (uses default input, output pins).

```
pt_shell> create_qtm_path_type -lib_cell AN2 -fanout 2 path1
```

The following command creates a path type path2 by using cell AN2, with an average fanout count of 2, and using pin 'A' as the starting point for the path type.

```
pt_shell> create_qtm_path_type -lib_cell AN2 -input_pin A -fanout 2 path2
```

The following command creates a path type path3 by using cell AN2, with an average fanout count of 2, using pin A as the start point of the arc, and pin Z as the endpoint of the arc.

```
pt_shell> create_qtm_path_type -lib_cell AN2 -input_pin A -output_pin Z
 -fanout 2 path3
```

Feedback

See Also

- create_qtm_constraint_arc
- create_qtm_delay_arc
- create_qtm_drive_type
- create_qtm_load_type
- create_qtm_model
- report_qtm_model
- save_qtm_model

---

create_qtm_port

Creates a quick timing model (QTM) port.

Syntax

string *create_qtm_port*

```
-type port_type
port_list
```

Data Types

```
port_type     string
port_list     list
```

Arguments

`-type port_type`

Specifies the type of port. The port can be one of the following types: input, output, inout, internal or clock. If you want a port to be a clock, define it as a clock port.

`port_list`

Specifies the list of QTM ports you want created.

Description

This command creates QTM port. You can create a single port or a list of ports with this command. The ports can be input, output, inout, internal or clock. Any port that constrains another port or has an edge triggered launch arc originating from it, has to be defined to be of the type 'clock'.

Feedback

Bused ports are also defined by giving the start and end index (A[0:5]). Bused ports cannot be internal.

To show the information about the current quick timing model, use the *report_qtm_model* command.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following example creates an input bused QTM port A[0:5].

```
pt_shell> create_qtm_port A[0:5] -type input
```

The following example creates a clocked QTM port CLK.

```
pt_shell> create_qtm_port CLK -type clock
```

The following example creates QTM output ports A, B, C, and D.

```
pt_shell> create_qtm_port {A B C D} -type output
```

The following example creates QTM internal pin D_int.

```
pt_shell> create_qtm_port D_int -type internal
```

See Also

- create_qtm_model
- remove_qtm_port
- get_qtm_ports
- report_qtm_model
- save_qtm_model
- set_qtm_port_drive
- set_qtm_port_load

---

create_rule

Creates a user-defined rule. Rules are checked to identify potential problems in constraints or in the design.

Syntax

Boolean *create_rule*

```
-name name
[-severity severity]
-message message_list
[-parameters parameter_list]
[-description description]
[-global]
-checker_proc user_defined_checker
```

Data Types

name	string
severity	string
message_list	list
parameter_list	list
description	string
user_defined_checker	string

Arguments

`-name` *name*

Specifies a unique name for the rule being created.

`-severity` *severity*

Specifies the severity of a violation of this rule. Valid values are: "fatal", "error", "warning" or "info". If not specified, the severity of the rule is "warning".

`-message` *message_list*

Specifies the predefined text fragments which, along with the parameter values, make up the message text for a violation. For example, a rule that prints a violation for one clock might have *message_list* such as {"Clock " " is created on a pin instead of a port."}. In this case, one parameter (the clock name) would be defined.

`-parameters` *parameter_list*

Specifies the names of parameters which are used in conjunction with the *message_list* to create the message text for a violation. For example, a rule that needs to print the clock name as a parameter might have *parameter_list* of {"clock"}.

`-description` *description*

The description is a string enclosed in double quotes which provides more information about the rule. It can describe why a violation is problematic, and suggest ways to address the problem.

`-global`

Indicates that this rule is intended to be checked in user-defined global checking procedures only. This means that the rule is scenario independent and is

checked only one time and not per scenario. Without *-global*, the rule is assumed to be checked in user-defined scenario checking procedure only.

`-checker_proc` *`user_defined_checker`*

Associate the user-defined rule with an existing user-defined checker procedure. The checker procedure is automatically invoked in the analyze_design command if the user-defined rule is enabled and included in the rules checked by the analyze_design command. A user-defined checker procedure can only issue violations of user-defined rules associated with the checker. If the user-defined rule is a scenario rule, the user-defined checkers is invoked during scenario checks. If the user-defined rule is a global rule, the user-defined checker is invoked during global checks. All user-defined rules associated with a user-defined checker procedure must be either scenario rules or global rules. A checker procedure cannot have both scenario rules and global rules associated with it.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Creates a rule for checking in the *analyze_design* command. A rule is used for formatting output related to potential problems in the design or in the constraints for a scenario. You can pass a user-defined rule checking procedure to the *analyze_design* command. In that procedure, you can call the *create_rule_violation* command to register a violation of a rule that you defined using the *create_rule* command.

To remove a user-defined rule, use the *remove_rule* command. To temporarily disable one or more rules, use the *disable_rule* command. You can specify properties for the rule using the *set_rule_property* command.

Examples

The following example creates a rule related to clocks with two parameters:

```
pt_shell> create_rule -name CLK_0001 -severity warning \\
    -message {"Clock " " has " " source(s) on inout ports"}  \\
    -parameters {"clock" "count"} \\
    -description "A clock defined on an inout port may become \\
        invalid when the port is used in output mode."
    -checker_proc my_checker
```

See Also

- create_rule_violation
- get_rules

Feedback

create_rule_violation

Creates a violation entry for the specified rule in user-defined checking procedures.

Syntax

string *create_rule_violation*

```
-rule rule_name
-parameter_values value_list
-details details_text
```

Data Types

```
rule_name      string
value_list     list
detail_text    string
```

Arguments

`-rule rule_name`

Specifies a unique name for the rule being violated.

`-parameter_values value_list`

Specifies a list of values which are used in conjunction with the *message_list* specified in *create_rule* to create the message text for a violation. For example, a rule that needs to print the clock name as a parameter might have parameter values of {"clock1"}.

`-details detail_text`

The details is a string enclosed in double quotes which provides more information about the rule violation. It can describe why a violation is problematic, and suggest ways to address the problem.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Creates a rule violation entry in user-defined checking procedures. It is only active within the *analyze_design* command. *create_rule_violation* should be used only in user-defined rule checking procedures.

Examples

The following example creates a rule violation of rule CAP_0001 in a user-defined checking procedure:

```
proc test_user_scenario_rule {} {
  create_rule_violation -rule CAP_0001 -parameter_values {"my_port"} \\
```

Feedback

```
    -details "some details here"
}
```

See Also

- create_rule

---

create_ruleset

Creates a set of related rules

Syntax

Boolean *create_ruleset*

```
-name name
rule_list
```

Data Types

```
name        string
rule_list   list
```

Arguments

`-name`

Specifies a unique name for the ruleset being created.

rule_list

The list of rules or rulesets to include in this ruleset. If a ruleset is specified, all rules from that ruleset are included.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Creates a named set of rules for this session. Rulesets can be passed to other commands which operate on rules.

It is an error to specify the same name as an existing ruleset. If you need to redefine an existing ruleset, first remove the old one with the *remove_ruleset* command and then create a new one containing the wanted rules.

Examples

The following example creates a ruleset named "user_clock_rules" and uses it in another command.

Feedback

```
pt_shell> create_ruleset -name user_clock_rules \\
    [get_rules "UDEF_CLK_*"]
pt_shell> enable_rule user_clock_rules
```

See Also

- create_rule
- get_rules
- remove_ruleset

---

create_scenario

Creates a scenario for multi-scenario analysis.

Syntax

```
status create_scenario
  [-name scenario_name]
  [-mode mode_name]
  [-corner corner_name]
  [-common_data file_list]
  [-common_variables variable_list]
  [-specific_data file_list]
  [-specific_variables variable_list]
  [-image image]
  [-affinity host_option_names]
  [-eco_data directory_name]
```

Data Types

```
scenario_name      string
mode_name          string
corner_name        string
file_list          list
variable_list      list
image              string
host_option_names  list
directory_name     string
```

Arguments

`-name scenario_name`

Specifies a single scenario. If you use the *-mode* and *-corner* options, PrimeTime automatically names the scenario *mode_name_corner_name*.

`-mode` *`mode_name`*

Specifies the mode comprising the scenario. You must use this option with the *-corner* option; do not use this option with the *-name* option. PrimeTime automatically names the scenario *mode_name_corner_name*.

`-corner` *`corner_name`*

Specifies the corner comprising the scenario. You must use this option with the *-mode* option; do not use this option with the *-name* option. PrimeTime automatically names the scenario *mode_name_corner_name*.

`-common_data` *`file_list`*

Lists files that contain commands and data that is common to more than one scenario. These files can be used to create a common image for all scenarios which share the same common data. Creating a common image causes a runtime and disk usage overhead, thus this option should only be used if it is required. This option cannot be used in conjunction with the *-image* option.

`-common_variables` *`variable_list`*

Lists variables that are common to more than one scenario. These variables are used in conjunction with the common data files to determine commonality between scenarios. The value of the variable currently at the manager are set on the worker before the sourcing of the common data files. This option cannot be used in conjunction with the *-image* option.

`-specific_data` *`file_list`*

Lists files that contain commands and data that is specific to this scenario. This data is not used on any other scenario or in the common image generation. This data that is specific to the scenario is sent directly to the worker and executed after the common design image is generated. This option can be used in conjunction with the *-image* option.

`-specific_variables` *`variable_list`*

Lists variables that are specific to this scenario. The value of the variable currently at the manager is set on the worker before the sourcing of the specific data files. This option can be used in conjunction with the *-image* option.

`-image` *`image`*

Allows a scenario to be created with a previously saved scenario image. The image can be an image generated within a standard PrimeTime session, an image generated with a save_session call from the DMSA manager or a current_image generated during a DMSA session. The *-specific_variables* and the *-specific_data* options can be specified in conjunction with the *-image* option. The specific variables are set after the image are loaded and then the specific data files are executed.

`-affinity` *`host_option_names`*

Lists the names of the host options from which worker processes must be selected to run the scenario. If no workers of any of the host_options specified in the affinity are online then the scenario cannot be part of the current session. If host options specified in the affinity are subsequently removed, then the affinity will also be updated to reflect the removal. If while running, all the workers of the host options specified in the affinity go offline/fatal, then un-expected tasks for the scenario will terminate due to a lack-of-resources and the scenario will be dropped from the current session.

`-eco_data` *`directory_name`*

Specifies the name of directory which is used to keep all ECO data populated by command '*write_eco_scenario_data*'.

Then, command '*report_eco_scenarios*' could use the data from this directory to dertermine *live* views based upon the number of processes specified by the '*set_host_options -num_processes*', for better timing coverage while with less memory resources.

Finally, command '*start_eco_scenarios*' could launch *live* views as *Distributed Multi Scenario Analysis (DMSA)* worker processes, and merges the timing and other data from *static* view scenarios with the *live* view scenarios.

Description

The *create_scenario* command creates a scenario based on the data specified by the arguments. A scenario can be created with a set of scripts and variables. Generally, scenarios only need to be defined with *-specific_data* and *-specific_variables*. When defined in this manner, all scenarios are independent and will each be created in parallel from the scripts and variables defined for them.

Any scenarios that have identical common data, may use the *-common_data* option to create a single baseline image. Creating a baseline image causes a runtime and disk usage overhead, thus this option should only be used if it is required, for example, if there are not enough worker processes to run all scenarios in parallel. In that case, the overhead of creating and restoring of the common image may be less than sourcing scripts for scenarios running sequentially.

Alternatively, a scenario can be created with a previously saved image.

This command does not actually start the analysis. The analysis starts when the first merged reporting or the *remote_execute* command is issued.

The *create_scenario* command is available only when you run PrimeTime in distributed multi-scenario analysis (DMSA) mode. The DMSA mode can be enabled by setting the variable *multi_scenario_enable_analysis* to *true*, or by invoking pt_shell with the *-multi_scenario* option.

You can run the *create_scenario* command any time in the DMSA flow before you run the *current_session* command.

If the *-affinity* option is specified, then when the current session is created, the scenario will only enter the session if at least one worker process launched from the host_options specified in the affinity are online.

Examples

The following example creates a scenario named scen1, which is described by the specific_s1.pt file.

```
pt_shell> create_scenario -name scen1 -specific_data {specific_s1.pt}
1
```

The following example creates a scenario named scen1, which is restored from the saved image stored in the directory *./scenarios/scen1*.

```
pt_shell> create_scenario -name scen1 -image ./scenarios/scen1
1
```

The following example creates scenarios named scen1 and scen2, which are described by the common_s1_s2.pt, specific_s1.pt and specific_s2.pt files.

```
pt_shell> create_scenario -name scen1 -common_data {common_s1_s2.pt} \\
            -specific_data  {specific_s1.pt}
1
pt_shell> create_scenario -name scen2 -common_data {common_s1_s2.pt} \\
            -specific_data  {specific_s2.pt}
1
```

In the following example, an iterative loop creates a set of scenarios. The same single.tcl script is used for all scenarios, but the corner variable distinguishes the scenarios.

```
        foreach corner {bc tc wc} {
 create_scenario \\
  -name ${corner} \\
  -specific_variables {corner} \\
  -specific_data "single.tcl"
}
```

The following example creates two host_options, one called BigHosts targeting the host called pthost1 and another called SmallHosts targeting pthost2. It then creates two scenarios, scen1 and scen2, each specifying their desired affinity.

```
        set_host_options -name BigHosts   -num_processes 1 pthost1
set_host_options -name SmallHosts -num_processes 1 pthost2
create_scenario -name scen1 -specific_data {specific_s1.pt} \\
```

Feedback

```
 -affinity {BigHosts}
create_scenario -name scen2 -specific_data {specific_s2.pt} \\
 -affinity {SmallHosts}
```

The following example creates two scenarios, along with additional ECO data directory for Hybrid timing view ECO fixing.

```
        create_scenario -name scen1 -specific_data {specific_s1.pt} \\
       -eco_data scen1/eco_data
create_scenario -name scen2 -specific_data {specific_s2.pt} \\
       -eco_data scen2/eco_data
```

See Also

- current_scenario
- remove_scenario
- report_multi_scenario_design
- set_host_options
- write_eco_scenario_data
- report_eco_scenarios
- start_eco_scenarios

---

create_si_context

Generates an SI context for selected blocks of the design. A top level design and full chip binary parasitics can also be generated.

Syntax

status *create_si_context*

```
[-include include_list]
[-instances instance_list]
[-parasitics_options para_options]
[-top_inst instance_name]
[-no_design_parasitics]
```

Data Types

```
include_list      list
instance_list     list
```

```
para_options      list
instance_name     string
```

Arguments

`-include` *include_list*

If this option is not specified, the context script will be generated for cross-talk. You can use this option to specify PrimeTime SI to generate the context scripts for cross-talk, noise or both. Allowed values are *xtalk* and *noise*. The generated context will be the same, only the script will be different.

`-instances` *instance_list*

Indicates that PrimeTime SI is to extract contexts for the given list of instances or for all the top level instances of the design.

`-parasitics_options` *para_options*

Specifies that a parasitic file has to be generated to be used to annotate the (block + context) design. The allowed values for the *para_options* options are *spef_format* and *sbpf_format*.

`-top_inst` *instance_name*

Specifies that PrimeTime should generate a top level design for the given instance. By default, PrimeTime generates top level design for the given design. This option can be used in conjunction with the *-instances* option to generate contexts and top level design for an arbitrary hierarchy. Note that all the instances provided in the *-instances* option must be the child instances of the top level instance provided in this option.

`-no_design_parasitics`

By default, this command also generates full chip parasitics in SBPF format. Use this option to indicate PrimeTime not to generate the SBPF file. This option should be used if your parasitic extraction tool generated SBPF.

Description

The *create_si_context* command generates contexts for all top level instances of the design or for the list of instances given in the *-instances* option. This context can be used to perform signal integrity analysis at the block level.

The effect of context on the block is the same as that of the top level and other blocks on the block. Hence, context enables accurate signal integrity analysis for block level analysis as if the block is being analyzed in the context of the full chip.

The command also generates a top level design and a top level script that can be used to perform chip level analysis after block level analysis is complete and respective ILMs are generated for each block. User can direct PrimeTime to generate contexts and top

level design for an arbitrary level of hierarchy instance using the *-top_inst* and *-instances* options.

The command creates one directory for each block (for which a context is being extracted) and outputs the context files inside the directory. Each directory contains a verilog context design (named wrapper.v), a TCL script (named wrapper.tcl) that can be used in performing block level analysis, a text file that contains aggressor annotation points (named wrapper.txt) that can be used later for annotating arrivals/slews for accurate block level analysis. The command also generates an infinite arrival script (named wrapper.pt.gz) that can be used to perform conservative block analysis in the context. The command can generate this arrival script for cross-talk, noise or both depending on the *-include* option. In addition, if the *-parasitics_options* option is given, a parasitic file (named wrapper.sbpf or wrapper.spef.gz) gets generated. Also at the level of this new directory, a top level verilog file (named design.v), a top level TCL script (named design.tcl) are also generated. In addition, if the *-no_design_parasitics* option is not given, a full chip binary parasitic file (named design.sbpf) gets generated.

The *create_si_context* command is affected by the *pt_ilm_dir* environment variable. The top design files and directories for individual blocks are created at the location given by the *pt_ilm_dir* variable.

Examples

The following example generates contexts for all top level designs along with context parasitics in SBPF and a top level design.

```
pt_shell> create_si_context -parasitics_options {sbpf_format}
```

The following example generates contexts for instances "blockA" and "blockB" along with context parasitics in SPEF.

```
pt_shell> create_si_context -instances {blockA blockB} \\
-parasitics_options {spef_format}
```

See Also

- pt_ilm_dir
- write_arrival_annotations
- create_ilm

---

create_supply_net

Creates a supply net defined for the specified power domain. The supply net is created in the logic hierarchy at the same scope as the specified power domain.

Feedback

Syntax

string *create_supply_net*

```
[-domain domain_name]
[-reuse]
[-resolve unresolved | parallel | one_hot | parallel_one_hot]
supply_net_name
```

Data Types

```
domain_name          string
supply_net_name      string
```

Arguments

`-domain domain_name`

Specify which power domain this supply_net is defined for. If specified, the power domain must exist. If used with the *-reuse* option, the *supply_net_name* must correspond to a supply net which is already domain dependent.

`-reuse`

Reuse an existing supply net instead of creating a new one. You can associate one supply net with several power domains. If this option is specified, the supply net must exist.

`-resolve unresolved | parallel | one_hot | parallel_one_hot`

Resolution mechanism that determines the state and voltage of the supply net. This is intended also for checking allowed connectivity of supply nets through switches. PrimeTime currently accepts and stores this option but does not use it.

supply_net_name

Specify the name of the supply net to be created. The name should be a simple (nonhierarchical) name.

In the scope of the specified power_domain, or in the scope of the current instance if no power_domain is specified, if there is a supply net, logical hierarchical net, or logical port with the same name, the *supply_net_name* cannot be created. One exception is that when you use the *-reuse* option, the name must be the same with another existing supply_net in the same scope, and no checking is applied in this situation.

This argument is required.

Description

The *create_supply_net* command enables you to create a supply net defined in the specified power domain. A supply net connects supply ports and pins.

Feedback

Each power domain has a primary power supply net and a primary ground supply net, and it could have several other supply nets. If the command succeeds, a supply net is created in the scope of power domain, or in the current scope, if no power domain is specified. It is currently neither primary power net nor primary ground net of the power domain. Use the *set_domain_supply_net* command to set it as the primary power/ground net.

Supply nets can be created either domain dependent, or domain independent (with or without the *-domain* option). If a supply net has been created domain independent, it cannot be changed to a domain dependent supply net by the use of the *-reuse* option in a later invocation of *create_supply_net*.

This command returns the full name of the supply net (from the current scope), if successful. It returns null string if it fails.

Examples

The following example creates a supply_net A_VDD in power_domain PD1, and also re-creates A_DD in another power_domain PD2 using the *-reuse* option:

```
prompt> create_power_domain PD1 -element INST1
PD1
prompt> create_power_domain PD2 -element INST2
PD2
prompt> create_supply_net A_VDD -domain PD1
A_VDD
prompt> create_supply_net A_VDD -domain PD2 -reuse
A_VDD
prompt> create_supply_net B
B
```

See Also

- create_power_domain
- report_supply_net
- set_domain_supply_net

---

create_supply_port

Creates a supply port in specified power domain, or in current scope if no power domain is specified.

Syntax

string *create_supply_port*

```
supply_port_name
[-domain domain_name]
[-direction in | out | inout]
```

Feedback

Data Types

```
supply_port_name     string
domain_name     string
```

Arguments

supply_port_name

Specifies the name of the supply port to be created. The name should be a simple (non-hierarchical) name. This argument can be used with the *-domain* option to help avoid hierarchical names.

In the scope of the specified *power_domain*, if there is a *supply_port*, logical port, or logical hierarchical net with the same name, the *supply_port* cannot be created.

This argument is required.

-domain *domain_name*

Specifies the *power_domain* in which this port defines a supply net connection point. The power domain must exist in the current scope.

-direction in | out | inout

Defines how state information is propagated through the supply network as it is connected to the port. If the port is an input port, the state information of the external supply net connected to the port is propagated into the domain. Likewise, for an output port, the state information of the internal supply net connected to the port is propagated out of the domain.

The default value is *in*.

Description

The *create_supply_port* command creates a supply port at the scope of the specified power domain, or at the current scope if no domain is specified. A supply port provides a connection point for the supply net.

This command returns the *supply_port* object upon success. It returns a null string if it failed.

Examples

The following example creates a supply port VN1 at the scope of the PD1 power domain:

```
prompt> create_power_domain PD1 -element INST1
PD1
prompt> create_supply_port VN1 -domain PD1
VN1
```

Feedback

See Also

- create_supply_net
- create_power_domain
- get_supply_ports

---

create_supply_set

Creates a set of supplies that can be used to define the power network. A supply set is created in the current logic hierarchy.

Syntax

string *create_supply_set*

```
supply_set_name
[-function {function_name net_name}]
[-update]
```

Data Types

```
supply_set_name        string
function_name          string
net_name               string
```

Arguments

supply_set_name

Specifies the name of the supply set to be created. The name should be a simple (non-hierarchical) name.

If a supply set with the same name exists in the current scope, the supply set is not created.

An exception to this rule is when you use the *-update* option. In this case, the supply set name must be the same as an existing supply set in the same scope.

This argument is required.

-function {function_name net_name}

Specifies the functional capability to which an actual supply net should be associated.

This argument takes two parameters. The first parameter is the name of the function, which is either *power* , *ground*, *pwell* or *nwell*. The second parameter is the name of the supply net that is in the same hierarchical scope of the supply set.

pwell and *nwell* are bias functions, and they are only accessible when design_attribute enable_bias is true. This attribute could be set using set_design_attribute.

This argument is required when the *-update* argument is used. Otherwise, it is optional.

`-update`

Instructs the tool to associate an actual supply net to the *power*, *ground*, *pwell* and *nwell* function of an existing supply set. It is an error if the supply set specified does not exist. Only one update is allowed for each functional of the supply set. If your update is not successful, the tool issues an error message. You can continue to update the functionality, until your update is successful.

This argument is optional. When the *-update* argument is used, the *-function* argument must also be used.

Description

The *create_supply_set* command creates a supply set in the current scope. A supply set eliminates the requirement of a supply net and supply ports in the design in the frontend tool. However, actual supply nets must be associated with the supply sets later on in the flow before physical synthesis is done.

A supply set supports the use of the following functions in the design:

- Power
- Ground
- Pwell
- Nwell

These functions, when unassociated with any supply net, can be used as supply nets for any purpose. They can be accessed by referencing them through the name of the supply set. To access the *power* function, you must specify *supply_set_name.power* and to access the *ground* function, you must specify *supply_set_name.ground*. Similarly, to access the *pwell* function, you must specify *supply_set_name.pwell* and to access the *nwell* function, you must specify *supply_set_name.nwell*.

The functions of a supply set can be used just as any other supply net in the UPF design, with no domain restrictions. However, domain restrictions are applicable when an *-update* is called. This means that the actual supply nets, being associated with a supply set, are checked for their definitions in the domains where the supply set functions have been used.

Feedback

Calling an update on an existing supply set means replacing the supply set functions with actual supply nets. After doing an *-update*, the actual supply net can be referenced through the supply set or by its actual name.

Examples

The following example creates a supply set named primary_sset in the current logical hierarchy.

```
pt_shell> create_supply_set primary_sset
primary_sset
```

The following example illustrates how a supply set function can be used.

```
pt_shell>set_design_attribute -attribute enable_bias TRUE -elements {.}
Information: Resolving '.' to the current scope '<top design>'.
1

pt_shell> create_supply_set primary_sset
primary_sset

pt_shell> create_power_domain PD_TOP
PD_TOP

pt_shell> set_domain_supply_net PD_TOP A \\
            -primary_power_net primary_sset.power \\
            -primary_ground_net primary_sset.ground \\
            -primary_power_net primary_sset.pwell \\
            -primary_ground_net primary_sset.nwell

1
pt_shell> set_retention ret_cstr -domain PD_TOP \\
            -retention_power_net primary_sset.power \\
            -retention_ground_net primary_sset.ground
1
pt_shell> set_isolation iso_cstr -domain PD_TOP \\
             -isolation_power_net primary_sset.power \\
             -isolation_ground_net primary_sset.ground \\
1
```

The following example illustrates how a supply set can be associated with actual supply nets.

```
pt_shell> create_supply_set primary_sset
primary_sset

pt_shell> create_power_domain PD_TOP
PD_TOP

pt_shell> set_domain_supply_net PD_TOP \\
           -primary_power_net primary_sset.power \\
           -primary_ground_net primary_sset.ground
```

```
1
pt_shell> set_retention ret_cstr -domain PD_TOP \\
            -retention_power_net primary_sset.power \\
            -retention_ground_net primary_sset.ground
1
pt_shell> set_isolation iso_cstr -domain PD_TOP \\
            -isolation_power_net primary_sset.power \\
            -isolation_ground_net primary_sset.ground \\
1
pt_shell> create_supply_net TOP_VDD -domain PD_TOP
TOP_VDD
pt_shell> create_supply_net TOP_VSS -domain PD_TOP
TOP_VSS
pt_shell> create_supply_set primary_sset \\
           -function {power TOP_VDD} \\
           -function {ground TOP_VSS} \\
           -update
`
1
```

See Also

- create_supply_net
- set_domain_supply_net
- set_retention
- set_isolation

---

create_testcase

Extracts a portion of the current design as a standalone testcase.

Syntax

string *create_testcase*

```
-paths path_list
   | -ports port_list
   | -endpoints pin_port_list
   | -cells cell_list
[-directory dir_name]
[-include_libs]
[-mim]
```

Data Types

```
cell_list        list
dir_name         string
path_list        list
```

```
pin_port_list     list
port_list         list
```

Arguments

`-paths` *path_list*

Extracts all cells along the specified timing paths, including startpoints and endpoints. The resulting testcase allows these timing paths to be reproduced.

In this mode, the clock path will not be included in the testcase unless the path is obtained with *-path full_clock_expanded*.

`-ports` *port_list*

Extracts fanout logic from input ports (including endpoints) and fanin logic to output ports (including startpoints).

In this mode, clock paths to sequential cells are always included.

`-endpoints` *pin_port_list*

Extracts fanin logic to endpoint pins/ports, including startpoints. The resulting testcase allows paths to these endpoints to be reproduced.

In this mode, clock paths to endpoints and their associated startpoints are always included.

`-cells` *cell_list*

Extracts the specified leaf cells, the nets driven by the cells, and the leaf cells and ports connected to the driven nets.

In this mode, clock paths are not included unless manually included in the cell list.

`-directory` *dir_name*

Specifies the output directory name. By default, the directory is named after the current design.

By default, the output directory is named after the current design.

`-include_libs`

Includes the required library files in the testcase directory, and modifies the script file to use them. This allows the testcase to be run in other design environments.

By default, the testcase references the required logic libraries in their current location. This is suitable for testcases that are used in the current design environment.

`-mim`

Keeps the same endpoint-fanout logic across multiple-instance module (MIM) instances.

When the *-endpoint* option specifies an endpoint inside an MIM instance, this option ensures that the logic is extracted consistently across the MIM instances. If this option is not used, the timing results of the testcase can differ from the original design.

This option requires that the *-endpoints* option be used to extract the testcase.

Description

The *create_testcase* command extracts a portion of the current design as a standalone testcase.

It creates a directory with the specified portion of the design, a constraints file that applies to that portion of the design, any detailed parasitics or SDF annotations used, and a run script for the testcase. Net fanout and cross-coupling are modeled so that the extracted testcase closely replicates the timing behaviors of the full design.

In a PrimeTime SI analysis, the aggressor stages with annotated aggressor timing windows are included.

The logic to be extracted must be specified by using one (and only one) of the following options: *-paths*, *-ports*, *-endpoints*, or *-cells*.

A timing-updated design is not required when using the *-ports* or *-endpoints* option. In this case, the aggressors use zero input pin slews and infinite windows. This is useful for reproducing crashes in the initial timing update.

Libraries are not copied to the test case directory by default, as they can require a lot of disk space. However, you can include them by specifying the *-include_libs* option.

Examples

The following script reads a full-chip design, then creates a small standalone testcase for a particular set of endpoints. The libraries are included in the testcase.

```
################################################
# FILE: run_full_design.tcl
#
#   Represents a typical script for STA and ECO in PrimeTime
#   Full design requires a 100 GB machine.
################################################

# STA
read_verilog ...
link_design CHIP
read_parasitics ...
read_sdf ...
```

Feedback

```
update_timing ...
report_timing ...

# Create smaller testcase design
set endpoints [get_pins ...]
create_testcase \
  -endpoints $endpoints \
  -directory my_testcase \
  -include_libs
exit
```

---

create_voltage_area

Creates a voltage area at the specified region for providing placement constraints of cells associated with the region.

Syntax

```
status create_voltage_area
   [-name voltage_area_name]
   [-coordinate coordinate_list]
   [-polygons polygon_coordinate_list]
   [-power_domains power_domain_list]
   [-guard_band_x guard_band_width]
   [-guard_band_y guard_band_width]
   cell_list
```

Data Types

```
voltage_area_name           string
coordinate_list             list
polygon_coordinate_list     list
power_domain_list           list
guard_band_width            double
cell_list                   list
```

Arguments

`-name voltage_area_name`

Specifies the name of the voltage area to be created. If there is a voltage area with the same name already, the voltage area cannot be created. The name *DEFAULT_VA* is reserved and cannot be used for a user-defined voltage area. The *-name* and *-power_domains* options are mutually exclusive, unless multiple power domains are specified.

`-coordinate coordinate_list`

Specifies a list of lower-left and upper-right coordinates of a voltage area. Specify the *coordinate_list* in the format of {llx1 lly1 urx1 ury1 ...}, where each set of {llx lly urx ury} defines a target rectangular placement area. The voltage

area geometry can be a rectangle or a rectilinear polygon. The unit is micron. The *-coordinate* and *-polygons* options are mutually exclusive.

`-polygons` *`polygon_coordinate_list`*

Specifies a list of polygons of a voltage area. Specify the *polygon_coordinate_list* in the format of {{{x1 y1} ... {xN yN} {x1 y1}} ...}, where each set of {x1 y1 ... xN yN x1 y1} defines the list of points of one polygon. The voltage area geometry can be a rectangle (five points) or a rectilinear polygon. The unit is micron. The *-coordinate* and *-polygons* options are mutually exclusive.

`-power_domains` *`power_domain_list`*

Creates a voltage area from one or more existing power domains. Unless mutliple power domains are specified, the name of the voltage area is identical to the name of the power domain. All hierarchical cells associated with the power domains are included in the voltage area. If multiple power domains are specified, the *-name* option is required.

`-guard_band_x` *`guard_band_width`*

Specifies the guard width in the horizontal direction. The guardband is the spacing along the boundary of the voltage area where cells cannot be placed because of the lack of power supply rails.

The value specified with this option must be a positive number.

`-guard_band_y` *`guard_band_width`*

Specifies the guard width in the vertical direction. The guardband is the spacing along the boundary of the voltage area where cells cannot be placed because of the lack of power supply rails.

The value specified with this option must be a positive number.

`cell_list`

Specifies a list of hierarchical cells that are contained in the voltage area. Multiple hierarchies are permitted, and one cell cannot be in two voltage areas. Leaf cells (or standard cells) cannot be specified in this option.

You must specify cell_list option when you specify the *-name* option, unless you use the *-power_domains* option to specify a single power domain.

Description

The *create_voltage_area* command creates a voltage area of specific geometry on the core area of the chip. The voltage area is associated with hierarchical cells. PrimeTime uses the voltage area information to insert ECO cells in correct locations if you use the *fix_eco_drc* or *fix_eco_timing* command with the *-physical_mode* option.

You can create a voltage area only after PrimeTime loads the physical database. To specify a file that contains *create_voltage_area* commands, use the *set_eco_options* command with the *-physical_constraint_file* option. This enables PrimeTime to load physical database, read the constraint file, and create voltage areas before starting to fix violations. The *create_voltage_area* commands in *read_sdc* are ignored.

Voltage areas can physically be completely nested; that is, one voltage area can lie completely inside another voltage area.

Voltage areas can also be logically nested; that is, one hierarchical cell can belongs to one voltage area, while one of its descendants belongs to another voltage area.

Voltage areas cannot be partially overlapped. The creation fails if you try to create a voltage area partially overlapping with an existing voltage area.

Examples

The following example creates a voltage area named V_AREA1 for hierarchical cell INST_1.

```
pt_shell> create_voltage_area -name V_AREA1 \\
          -coordinate {100 100 200 200} INST_1
```

The following example creates a voltage area from an existing power INST. domain:

```
pt_shell> create_voltage_area -power_domains INST \\
          -coordinate { 215 215 350 350 }
```

The following example creates a voltage area by using the *-polygons* option:

```
pt_shell> create_voltage_area -power_domains INST \\
          -polygons {{100 100} {300 100} {300 200} {200 200} {200 300} \\
          {100 300} {100 100}}
```

The following example creates a voltage area with disjoint polygons by using the *-polygons* option:

```
pt_shell> create_voltage_area -power_domains INST \\
          -polygons {{{100 100} {300 100} {300 200} {200 200} {200 300}
 \\
          {100 300} {100 100}} {{400 400} {500 400} {500 500} {400 500}
 \\
          {400 400}}}
```

The following example creates a voltage area named V_AREA2 from two existing power domains INST and MULT:

```
pt_shell> create_voltage_area -power_domains "INST MULT" \\
          -coordinate { 200 200 450 450 }
```

Feedback

See Also

- create_placement_blockage
- fix_eco_drc
- fix_eco_timing
- set_eco_options
- write_changes

---

create_waiver

Creates a violation waiver. Waivers can conditionally suppress violations of an enabled rule.

Syntax

Boolean *create_waiver*

```
[-name name]
[-design design]
[-scenario scenario]
-rule rule_name
[-condition condition]
[-not_condition condition]
[-cells cell_list]
[-all_scenarios]
[-comment comment]
```

Data Types

```
name          string
design        string
scenario      string
rule_name     string
condition     list
cells         list
comment       string
```

Arguments

`-name name`

Specifies a name for the waiver being created. If there is an existing waiver with this name, it is overwritten. If no name is given, a unique name is automatically generated for the waiver.

`-design` *design*

Specifies the design to which the waiver is applied. If no design is given, the waiver is created for the current design. Expects linked design to be passed.

`-scenario` *scenario*

Specifies the scenario to which the waiver is applied. If no scenario is given, the waiver is created for the current scenario.

`-rule` *rule_name*

Specifies the rule. The waiver conditionally suppresses violations of the given rule.

`-condition` *condition*

There can be multiple *-condition* options. Each *-condition* specifies one requirement for the violation to be suppressed. *condition* is a list, where the first element is the name of a rule parameter, and all remaining elements are collections of netlist/constraint objects or strings. A *-condition* option is satisfied if the value of the violation parameter falls in one of the collections or strings. A violation is suppressed only if all the *-condition* and *-not_condition* options are satisfied.

`-not_condition` *condition*

There can be multiple *-not_condition* options. Each *-not_condition* specifies one requirement for the violation to be suppressed. *condition* is a list, where the first element is the name of a rule parameter, and all remaining elements are collections of netlist/constraint objects or strings. A *-not_condition* option is satisfied if the value of the violation parameter does not fall in any of the collections or strings. A violation is suppressed only if all the *-condition* and *-not_condition* options are satisfied.

`-cells` *cell_list*

List of cells for instance based waivers. The list can contain hierarchical or leaf cells. For leaf cells, the violations that affect only such instances are waived. For hierarchical cells, the violations that affect only objects inside or on the hierarchical instance are waived.

`-all_scenarios`

Use this option in conjunction with *-cells* option to indicate that the instance based waiver should apply to all the scenarios and not just the scenario specified by the "-scenario" option.

`-comment` *comment*

A comment on the waiver (up to 1024 characters).

Feedback

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Creates a violation waiver for the given rule. A violation waiver suppresses a violation if the violation satisfies all *-condition* and *-not_condition* conditions. Multiple waivers can be created on one rule. A violation is suppressed if it meets the requirements of at least one waiver.

If the rule is scenario-dependent, the waiver is active when checks are performed in the scenario specified by the "-scenario" option, or the current scenario if no scenario is specified; if the rule is scenario-independent, the waiver is active in scenario-independent checks only.

To remove a violation waiver use the *remove_waiver* command.

Examples

The following example creates a waiver for CLK_0003 violations if the *clock* parameter is CLK1 or CLK2, and if the *pin* parameter is not U1/A or U1/B:

```
pt_shell> create_waiver -name my_waiver1 -rule CLK_0003 \\
    -condition [list "clock" [get_clocks {CLK1 CLK2}]] \\
    -not_condition [list "pin" [get_pins U1/A] [get_pins U1/B]] \\
    -comment "waiver example"
```

The following example creates a waiver for all objects inside block instance hier_inst. Also, it specifies that the waiver should be applied to all scenarios.

```
pt_shell> create_waiver -name my_waiver1 -cells [get_cell hier_inst] \\
    -all_scenarios \\
    -comment "waiver example2"
```

When creating a waivers for block-to-top and SDC-to-SDC rules, *current_design* and *current_scenario* can be used to switch the context under which the object is to be retrieved. The following example shows how to create a waiver for B2T_CLK_0012 rule.

```
pt_shell> create_waiver -rule B2T_CLK_0012 \\
    -condition [list "blk_object" [current_design BLOCK; get_pins
 zGate/A]] \\
    -condition [list "blk_object_type" "pin"] \\
    -condition [list "top_object" [current_design TOP; get_pins
 b1/zGate/A]] \\
    -condition [list "top_object_type" "pin"]
```

The following example shows how to create a waiver for the S2S_CLK_0001 rule.

```
pt_shell> create_waiver -rule S2S_DIS_0001 \\
    -condition [list "object" [current_design S2S_CLK_0001; get_pins
 ck2_i2]] \\
    -condition [list "object_type" "pin"] \\
```

Feedback

```
    -condition [list "scenario1" "set2"] \\
    -condition [list "scenario2" "set1"]
```

See Also

- remove_waiver
- write_waiver

---

current_design

Sets or returns the current design.

Syntax

string *current_design*

`[design_name]`

Data Types

design_name string

Arguments

design_name

Specifies the working or focal design for many PrimeTime commands. If you omit the *design_name* option, the *current_design* command returns a collection containing the current design. If the *design_name* option specifies a design that cannot be found, an error is issued and the working design remains unchanged.

Description

This command sets or returns the working design, which is the focus of many PrimeTime commands. Before *link_design current_design* won't return anything. Without arguments, the *current_design* command returns a collection containing the current working design. Together, the *current_design* and *current_instance* commands define the focus for many PrimeTime commands.

To display designs currently available in PrimeTime, use the *list_designs* command.

Examples

The following example uses the *current_design* command to show the current context and to change the context from one design to another.

```
pt_shell> current_design
{"TOP"}

pt_shell> list_designs
```

```
Design Registry:
    ADDER              /designs/dbs/my_design.db:ADDER
    FULL_ADDER         /designs/dbs/my_design.db:FULL_ADDER
    FULL_SUBTRACTOR    /designs/dbs/my_design.db:FULL_SUBTRACTOR
    HALF_ADDER         /designs/dbs/my_design.db:HALF_ADDER
    HALF_SUBTRACTOR    /designs/dbs/my_design.db:HALF_SUBTRACTOR
    SUBTRACTOR         /designs/dbs/my_design.db:SUBTRACTOR
 *  TOP                /designs/dbs/my_design.db:TOP

pt_shell> current_design ADDER
{"ADDER"}

pt_shell> current_design
{"ADDER"}
```

The *current_design* command can be used as a parameter to other *pt_shell* commands. In the following example, the *remove_design* command is used to delete the working design from *pt_shell*.

```
pt_shell> current_design
{"TOP"}
pt_shell> remove_design [current_design]
Removing design 'TOP'...
1
pt_shell> current_design
Error: Current design is not defined. (DES-001)
pt_shell>
```

See Also

- current_instance
- list_designs

---

current_instance

Sets the current working cell instance, which enables subsequent commands to be used relative to that instance.

Syntax

string *current_instance*

```
[instance]
```

Data Types

```
instance     string
```

Feedback

Arguments

instance

Specifies the current working cell instance to be used as the focus for subsequent commands.

Description

The *current_instance* command specifies the working cell instance. This command differs from the *current_design* command, which specifies the working design and then sets the current instance to the top level of the current design. The combination of the current design and current instance defines the focus for many PrimeTime commands.

The *current_instance* command allows you to traverse the design hierarchy similar to the way the shell *cd* command traverses the file system hierarchy. The *current_instance* command operates with a variety of *instance* arguments:

- If you do not specify an *instance* argument, the command focus returns to the top level of the hierarchy.
- If *instance* is one period (.), the command returns the current instance, and no change is made.
- If *instance* is two periods (..), the current instance moves up one level in the design hierarchy.
- If *instance* is a valid cell (or cell name) at the current level of hierarchy, the command focus moves down to that level of the design hierarchy.

Most commands that return port and pin objects consider the current instance. By default the *get_ports* command always returns ports of the top-level design. To enable the *get_ports* command to return pins of the current instance when applicable, set the *port_search_in_current_instance* variable to *true*. See EXAMPLES for details.

Examples

The following example uses *current_instance* to move up and down the design hierarchy.

```
pt_shell> current_design TOP
{"TOP"}

pt_shell> current_instance U1
U1

pt_shell> current_instance .
U1

pt_shell> current_instance U3
U1/U3
```

```
pt_shell> current_instance ../U2
U1/U2

pt_shell> current_instance
Current instance is the top-level of design 'TOP'.
```

The following example shows the use of more complex instance paths with multiple hierarchy levels separated by a slash:

```
pt_shell> current_instance U1/U3/U5
U1/U3/U5

pt_shell> current_instance ../../U2/U4
U1/U2/U4
```

The following example shows how the *get_ports* command operates within a current instance:

```
pt_shell> current_instance
Current instance is the top-level of design 'TOP'.

pt_shell> get_ports *
{"CLK", "DIN[0]", "DIN[1]", "DOUT[0]", "DOUT[1]"}

pt_shell> current_instance reg0
reg0

pt_shell> get_ports *
{"CLK", "DIN[0]", "DIN[1]", "DOUT[0]", "DOUT[1]"}

pt_shell> get_attribute [get_ports *] object_class
{"port", "port", "port", "port", "port"}

pt_shell> set_app_var port_search_in_current_instance true
true

pt_shell> get_ports *
{"reg0/clk", "reg/d", "reg/q"}

pt_shell> get_attribute [get_ports *] object_class
{"pin", "pin", "pin"}
```

See Also

- all_instances
- current_design
- list_designs

Feedback

- query_objects
- port_search_in_current_instance

---

current_power_rail

Sets or gets the power rails in a multi-rail design to be included in power analysis. As the default (if not specified), all the power rails in the design are included in power analysis. This command has no effect on single rail design.

When the variable power_enable_multi_rail_analysis is true, specifes the power rails that will be included in reporting. When false, sets the specific power rails to be included in the power analysis.

Syntax

int *current_power_rail*

`[rail_names]`

Data Types

rail_names	string

Arguments

rail_names

Specifies the names of the power rails in the design to be included in power analysis or reporting. If the *rail_names* option is not specified, the *current_power_rail* command returns a collection of power rails being included in power analysis. If the *rail_names* option refers to a power rail that is not defined in the design, an error is issued and the current rail settings remains unchanged.

Description

Sets or gets the power rails in a multirail design to be included in power analysis or in reporting. Without arguments, the *current_power_rail* command returns a collection of power rails being included in power analysis.

The command behavior depends on the power_enable_multi_rail_analysis variable. When false, (the default); concurrent multi-rail analysis is disabled; and power for all the power rails in the design are calculated. This command controls which power rails are to be included in the power analysis. When specified, only the power dissipation on the listed power rails is calculated and reported. To report power consumption per power rail, the command must be applied for each power rail, and the power reanalyzed per power rail.

When the power_enable_multi_rail_analysis variable is true, power per power rail is calculated, and the command controls the power reporting and attribute values. When specified, the power dissipation on the listed power rails is accessible via the get_attribute

and report_attribute commands, as well as the report_power command. To access the power attribute values per power rail, the command must be applied for each power rail, before issueing the get_attribute or report_attribute commands.

Before using this command, you define the design power rails by using the *create_power_rail_mapping* command. The *report_power_rail_mapping* command can be used to show the defined design power rails, library power rails, and rail mapping information.

The following power analysis results will be affected by this command in a multi-rail design:

```
- average power report
- time-based power report
- power related attributes: total_power
                            dynamic_power
                            internal_power
                            switching_power
                            leakage_power
                            x_transition_power
                            glitch_power
                            peak_power
```

Use the *current_power_rail all* command to resume the default behavior, which includes all the power rails in the design.

Peak_power and peak_time per rail are only accessible when power_enable_multi_rail_analysis is false. When concurrent multi-rail analysis is performed, only the peak power and peak time for the all the supply nets is accessible. Per rail the peak power attributes are available only when concurrent multi-rail analysis is disabled.

Examples

The following example shows generating rail-based power analysis results. The first *report_power* command will generate the power analysis results for the part of design with supply voltage of VDD1 and VDD2. The second *report_power* command will generate the results for the part of design which is supplied by VDD1.

```
pt_shell> create_power_rail_mapping VDD1 -cells block1
pt_shell> create_power_rail_mapping VDD2 -cells block2
pt_shell> current_power_rail {VDD1 VDD2}
pt_shell> report_power
pt_shell> current_power_rail VDD1
pt_shell> report_power
```

Feedback

See Also

- create_power_rail_mapping
- report_power_rail_mapping
- update_power
- report_power
- report_power_calculation

---

current_scenario

Selects a subset of the scenarios in the current session for analysis.

Syntax

status *current_scenario*

```
scenario_list
[-all]
```

Data Types

```
scenario_list        list
```

Arguments

scenario_list

Specifies a list of scenarios to analyze in the current session . If you specify a scenario that exists but is not in the current session, the command issues an error and does not change the command focus. The list can also contain patterns with the asterisk (*) and question mark (?) wildcard symbols to specify multiple scenarios. If you use wildcards symbols, the scenarios selected are automatically restricted to come from the scenarios in the current session.

-all

Selects all scenarios in the current session for analysis.

Description

The *current_scenario* command returns a collection containing all the scenarios in command focus. This command is only available if you invoke the *pt_shell* command with the *-multi_scenario* option.

Before you run the *current_scenario* command, you need to create scenarios by running the *create_scenario* command.

In multi-scenario analysis, after you create scenarios, use the *current_session* command to focus on a set of scenarios for analysis. Next, focus the analysis on a specific subset of the scenarios in the current session by running the *current_scenario* command; subsequent commands will apply only to the specified scenarios.

To determine which scenarios are in command focus, run the *report_multi_scenario_design* command with the *-session* option. If you run the command at remote workers by *remote_execute* command, it returns the name of the scenario executing the command. In this mode, the options of the command are disabled.

Examples

In the following example, three scenarios named scen1, scen2, and scen3 are created.

- scen1 is created using common_s1.pt and specific_s1.pt
- scen2 is created using common_s2.pt and specific_s2.pt
- scen3 is created using common_s3.pt and specific_s3.pt

The scenarios scen1, scen2 and scen3 are selected for the current session, which is then displayed. Notice that all three scenarios are in command focus.

The analysis is focused on only scen2 and scen3 using the *current_scenario* command and the session is displayed again. Notice that scen1 is now only in session focus but scen2 and scen3 are in command focus. That is, only scen2 and scen3 will be considered for analysis.

```
1
pt_shell> create_scenario -name scen1 -common_data {common_s1.pt} \\
-specific_data  {specific_s1.pt}
1
pt_shell> create_scenario -name scen2 -common_data {common_s2.pt} \\
-specific_data  {specific_s2.pt}
1
pt_shell> create_scenario -name scen3 -common_data {common_s3.pt} \\
-specific_data  {specific_s3.pt}
1
pt_shell> current_session {scen1 scen2 scen3}
{"scen1", "scen2", "scen3"}
pt_shell> report_multi_scenario_design -session

****************************************
Report : multi_scenario_design
 ...
****************************************

Session details
-------------------------------------------------------

Number of scenarios in current session: 3
```

```
        Focus           Scenario  (Image provider)

        Command         scen1   (scen1)
        Command         scen2   (scen2)
        Command         scen3   (scen3)

1
pt_shell> current_scenario {scen2 scen3}
{"scen2", "scen3"}
pt_shell> report_multi_scenario_design -session

****************************************
Report : multi_scenario_design
 ...
****************************************


Session details
-------------------------------------------------------

Number of scenarios in current session: 3

        Focus           Scenario  (Image provider)

        Session         scen1   (scen1)
        Command         scen2   (scen2)
        Command         scen3   (scen3)

1
```

In the following example, two scenarios named scen1 and scen2 are in focus. The *remote_execute* command is used to save PrimeTime sessions for scenarios scen1 and scen2 in directories named /u/ptms/sessions/sess_scen1 and /u/ptms/sessions/sess_scen2, respectively.

```
pt_shell> sh mkdir /u/ptms/sessions
pt_shell> remote_execute
 {save_session /u/ptms/sessions/sess_[current_scenario]}

Start of Manager/Worker Task Processing
-----------------------------------
Started    : Command execution in scenario 'scen2'
Started    : Command execution in scenario 'scen1'
Succeeded  : Command execution in scenario 'scen2'
Succeeded  : Command execution in scenario 'scen1'
-----------------------------------
End of Manager/Worker Task Processing

1
pt_shell> ls /u/ptms/sessions
pt_shell>
```

Feedback

See Also

- create_scenario
- current_session
- remove_scenario
- report_multi_scenario_design

---

current_session

Selects a set of scenarios for analysis.

Syntax

string *current_session*

```
[scenario_list]
[-all]
```

Data Types

```
scenario_list     list
```

Arguments

scenario_list

Brings the specified scenarios into focus for analysis. To specify multiple scenarios, you can use the * and *?* wildcard symbols.

`-all`

Brings all defined scenarios into focus for analysis.

Description

This command is available only if you invoke the pt_shell with the *-multi_scenario* option.

To set the current session, you need the following resources:

- At least one PrimeTime license is available for usage; since the manager can share its license with workers this criterion is always met; to query additional license that are currently checked out and can be shared with use the *report_multi_scenario_design* commadn passing *-license* option.
- At least one host option has been defined using the *set_host_options* command; to query the host options that have been created use the *report_host_usage* command.
- At least one worker process has been brought online using the *start_hosts* command; to query the workers that have come online, use the *report_host_usage* command.

Feedback

After all required resources are available, and the scenarios are created, you must select a set of scenarios to analyze by using the *current_session* command. Calling this command with a list of scenarios brings those scenarios into focus for the current session (session focus) and into focus for receiving commands (command focus) from the manager. The current session can be set with all or a subset of the created scenarios. If a scenario selected has an affinity specification, then the scenario will only be allowed enter the session if at least one worker process is online from the host options in its affinity specification.

The *current_session* command returns a collection containing all the scenarios in the current session.

Calling the *current_session* command resets all scenarios to the way they were when created by the *create_scenario* command. As part of executing the subsequent distributed command, the scenarios in command focus in the session will be rebuilt from the scripts or images used to originally define them.

If the *multi_scenario_license_mode* variable is set to '*core*' then there is a requirement that all the scenarios in the session are using the same design. When each scenario loads its design, if it detects its design is different to any other scenario in the session, then the scenario will error out, the command executing at the manager will error out and the current session will terminate with an error.

Examples

In the following example, a configuration and three scenarios named scen1, scen2 and scen3 are created. The current session is set such that scen2 and scen3 are in session and command focus.

- scen1 is created using common_s1.pt and specific_s1.pt
- scen2 is created using common_s2.pt and specific_s2.pt
- scen3 is created using common_s3.pt and specific_s3.pt

The scen2 and scen3 scenarios are brought into focus.

```
pt_shell> create_scenario -name scen1 -common_data {common_s1.pt} \\
            -specific_data  {specific_s1.pt}
1
pt_shell> create_scenario -name scen2 -common_data {common_s2.pt} \\
            -specific_data  {specific_s2.pt}
1
pt_shell> create_scenario -name scen3 -common_data {common_s3.pt} \\
            -specific_data  {specific_s3.pt}
1
pt_shell> current_session {scen2 scen3}
{"scen2", "scen3"}
```

Feedback

See Also

- current_scenario
- remove_scenario
- report_host_usage
- report_multi_scenario_design
- multi_scenario_license_mode

d

date

Returns a string containing the current date and time.

Syntax

string *date*

Description

The *date* command generates a string containing the current date and time, and returns that string as the result of the command. The format is fixed as follows:

```
ddd mmm nn hh:mm:ss yyyy
```

Where:

```
ddd is an abbreviation for the day
mmm is an abbreviation for the month
nn is the day number
hh is the hour number (24 hour system)
mm is the minute number
ss is the second number
yyyy is the year
```

The *date* command is useful because it is native. It does not fork a process. On some operating systems, when the process becomes large, no further processes can be forked from it. With it, there is no need to call the operating system with *exec* to ask for the date and time.

Examples

The following command prints the date.

```
prompt> echo "Date and time: [date]"
Date and time: Thu Dec  9 17:29:51 1999
```

Feedback

debug_script

Debug a script using the TclDevKit Tcl debugger.

Syntax

string *debug_script* script [*port*] [*hostname*]

Arguments

script

This specifies the Tcl script(s) to be debugged. This script will be sourced and instrumented for debugging.

port

Specifies the port number that the debugger is listening on for a remote debugging application. By default the debugger will be launched on an available port.

hostname

Specifies the host where Prodebug is already running. This is used to connect to an existing remote Prodebug session.

::env(SNPS_TCLPRO_HOME)

This Tcl variable is used to specify the root of the TclPro installation. It is initialized from the environment variable SNPS_TCLPRO_HOME in the user's environment.

USING THE PACKAGE

The *debug_script* command is in the snpsTclPro Tcl package, and all of the commands in this package are defined in the namespace snpsTclPro. The way to use this command is to load the package, which will import the commands it exports into the global namespace. This is shown below.

```
shell> package require snpsTclPro
1.0
```

These commands can be added to the .synopsys setup file for the system, user, or current directory, or the commands can be typed when the pacakge is needed.

Description

The *debug_script* command simplifies the use of the TclDevKit Tcl debugger available from ActiveState (http://www.activestate.com). First it ensures a connection to the Prodebug debugger either by connecting to a running debug session (given a *hostname* and *port*), or by launching the debugger on *localhost* using an available port, and then connecting to it.

If there is already an active connection to the debugger then that debugger will simply be used.

Once the connection to the debugger is set up, the specified script will be sourced and debugged. The default behavior for the debugger is to stop at the first line of the source'd script.

The debugger must be run using a "remote" debugging project if you want to connect the Synopsys tools to a running prodebug session. The port number in use by the debugger can be viewed via the Application tab of the "File->Project" Settings dialog.

Examples

```
# Launch the debugger and then debug the script myscript.tcl.
shell> debug_script myscript.tcl
Selected port 2576 on host localhost
launching prodebug

# now debug another script using the same session
shell> debug_script anotherScript.tcl

# debug myscript.tcl, connecting to the specified debugger session or
# creating one if the connection fails.
shell> debug_script myscript.tcl 2576 localhost
```

CAVEATS

The debugger currently kills the process that started the debugger when you exit, despite the preferences being set differently in the debugger.

When the Synopsys tool has spawned a debugger, that debugger must be exited before the Synopsys tool can exit. If this is not the case then the Synopsys tool will hang on exit, until the debugger is exited.

The *debug_script* command waits for the debug process to start up, before it tries to connect to the debugger. If *debug_script* times out before the debugger is started due to machine or network loading, simply close the debugger that was launched, and then increase the timeout setting with the command *set_debugger_start_timeout*, and then re-run the debugger. The *set_debugger_start_timeout* command takes a single argument which is the value of the timeout in miliseconds, and the default value for the timeout is 10000.

See Also

- check_script

Feedback

define_cell_alternative_lib_mapping

Specifies which library file to use from an exact-match-only scaling library group for timing analysis of named cell instances.

Syntax

string *define_cell_alternative_lib_mapping*

```
library_file_name
-cells cell_list
[-sms_scenarios sms_scenarios_list]
```

Data Types

```
library_file_name     string
cell_list             list
sms_scenarios_list    collection
```

Arguments

library_file_name

Specifies the scaling group library file to associate with the cell. The library must be in a scaling library group previously defined by the *define_scaling_lib_group* command using the *-exact_match_only* option. For both leaf and hier cells, the list of library files supported.

-cells *cell_list*

Specifies a list of cell instances that will use the named library for timing analysis. They can be hierarchical or leaf-level cells.

-sms_scenarios *sms_scenarios_list*

Specifies a collection of SMS scenarios for which the command is applied. This collection is created by *get_sms_scenarios*.

Description

The *define_cell_alternative_lib_mapping* command allows you to associate cell instances with exact-match scaling libraries. Unlike the *link_path_per_instance* variable, you can apply updated cell/library associations without relinking the design.

Use this command after the *define_scaling_lib_group* command and before the *update_timing* or *report_timing* command.

Leaf cells can be associated with a single library or multiple libraries ,if its associated with multiple library then will pick the first library. Hierarchical cells can be associated with a list of libraries. For each leaf cell in the hierarchy, the first library containing the corresponding library cell is used.

Feedback

Specifications applied to leaf cells have the highest precedence. Specifications applied to lower hierarchy levels take precedence over specifications applied to higher hierarchy levels.

Make sure that the operating conditions and rail voltage set on the named cells match those of the named library file.

Examples

The following script creates an exact-match-only scaling library group containing two libraries and directs the tool to use a specific library for cells U1 and U2.

```
set link_path "* my_linked_lib.db"
read_verilog my_design.v
link_design my_design
define_scaling_lib_group -exact_match_only {my_linked_lib.db
 my_alt_lib.db}
define_cell_alternative_lib_mapping my_alt_lib.db -cells {U1 U2}
update_timing
report_timing
```

See Also

- report_delay_calculation
- report_lib_groups
- define_scaling_lib_group
- set_operating_conditions
- set_rail_voltage

---

define_derived_user_attribute

Create an attribute defined in Tcl

Syntax

string *define_derived_user_attribute* -type *data_type*

[-user_type *type_name*] -classes *class_list* -name *attribute_name* [-info *attribute_info*] -get_command *get_cmd* [-set_command *set_cmd*] [-remove_command *remove_cmd*] [-subscript_command *sub_cmd*]

```
string data_type
string type_name
list  class_list
string attribute_name
string attribute_info
string get_cmd
```

```
string set_cmd
string remove_cmd
string sub_cmd
```

Arguments

`-type data_type`

The type of the attribute (Values: boolean, double, int, string, collection)

`-user_type type_name`

User type name. Some applications use the user_type to generate improved dialogs for setting attribute values. If this is not specified the type name is derived from the specified attribute type.

`-classes class_list`

Define attribute on these classes

`-name attribute_name`

Name of the attribute

`-info attribute_info`

Sort description of the attribute value.

`-get_command get_cmd`

Tcl command to retrieve the attribute value.

`-set_command set_cmd`

Tcl command to set the attribute value. The specified script is evaluated when the *set_attribute* command is executed.

`-remove_command remove_cmd`

Tcl command to remove the attribute value. The specified script is evaluated when the *remove_attribute* command is executed.

`-subscript_command sub_cmd`

Define a subscripted attribute. Specifying this option causes the defined attribute to be subscripted. The command should return the valid set of subscripts on this object.

Description

The *define_derived_user_attribute* command defines an attribute on a class of objects. The value of that attribute is computed by evaluating the script provided by the *-get_command* option.

Feedback

All of the commands supplied for the attribute may reference special keys. The supported keys are the following:

%object

The object to operate on. This key is supported in all commands.

%subscript

This is only valid when a subscripted attribute is created. This key is replaced by the specified subscript in the get, set and remove commands.

%value

This key is only valid in set commands. It contains the value to set the attribute to.

All of the commands supplied for the attribute define an anonymous proc context. This means any variables you refer to in these scripts are scoped locally. The anonymous proc is created in namespace used to evaluate the *define_derived_user_attribute* command. To refer to variables from that enclosing namespace use the *variable* command. You may also refer to global variables via the *global* command or by use the "$::varName" syntax.

If the value returned by the *get_command* is the empty string, then the attribute is not set on this object. This may be tested by the defined and undefined operator in the *filter_collection* command. If the value is not the empty string, the returned value is converted into the specified type. If the string cannot be converted into the type, it is considered an error.

Examples

Declare an attribute called user_tag on cells. The attribute stores attribute values into a Tcl dict keyed on the full_name of the cell. The dict is maintained in a namespace:

```
namespace eval user_tag_ns {
  # Map from name - val
  variable vals;                # Dictionary name->value

  define_derived_user_attribute -classes cell -name user_tag -type int \\
    -get_command {
      variable vals
      if {[catch {set val [dict get $vals [get_attribute %object
 full_name]]} val]} {
        return "";          # Unset.
      }
      set val
    } \\
    -set_command {
      variable vals
      set name [get_attribute %object full_name]
      dict set vals $name %value
```

Feedback

```
    } \\
     -remove_command {
       variable vals
       set name [get_attribute %object full_name]
       dict unset vals $name
    }
  }
```

See Also

- get_attribute
- undefine_derived_user_attribute
- get_defined_attributes
- filter_collection

---

define_design_mode_group

Defines a design mode group with a set of design modes.

Syntax

string *define_design_mode_group*

```
[-group_name name]
mode_list
```

Data Types

```
name            string
mode_list       list
```

Arguments

`-group_name name`

Specifies the name of the design mode group. By default, if no design mode group is specified, a default design mode group is created and named "default".

mode_list

Specifies a list of design modes to be created and included in the design mode group.

Description

The *define_design_mode_group* command defines a set of design modes to be placed in a design mode group for your design. You select one of the design modes to be the active mode using the *set_mode -type design* command. There are only two valid states for a design mode group. The default state of all modes enabled or the state of one mode

enabled and all others disabled. Setting one of the modes to be active automatically sets the rest of the modes in the group inactive (disabled).

You can map cell modes or paths to a design mode using the *map_design_mode* command. When you select the active design mode with the *set_mode -type design* command, the modes of the cells specified for that design mode are also active. Any paths mapped to disabled design modes automatically become false paths.

Unlike design modes, which are user-specified, cell modes are predefined, and are in the library. You can find out what cell modes are available using the *report_mode* command. For more information about cell modes, see the manual page for the *set_mode* command.

To see a report of the design modes specified for the current design, use the *report_mode -type design* command.

To remove design modes, use the *remove_design_mode* command. Removing a design mode also removes any cell mode or path specifications previously mapped to the design mode using the *map_design_mode* command.

Examples

The following example defines two design modes for the current design: SYSTEM and TEST.

```
pt_shell> define_design_mode_group {SYSTEM TEST}
```

The following example defines two independent design mode groups. The first command defines a design mode group named RW that contains design modes READ and WRITE. The second command defines a design mode group named LT that contains design modes LATCH and TRANSPARENT.

```
pt_shell> define_design_mode_group -group RW {READ WRITE}
pt_shell> define_design_mode_group -group LT {LATCH TRANSPARENT}
```

The following command enables the design mode READ and disables the design mode WRITE, because READ and WRITE belong to the same design mode group RW.

```
pt_shell> set_mode -type design READ
```

See Also

- remove_design_mode
- map_design_mode
- report_mode
- reset_mode
- set_mode

Feedback

define_flow_summary_segment

Define a custom event in flow summary report which is not a PrimeTime command.

Syntax

status *define_flow_summary_segment* [-start] [-end] [-name *event_name*]

Data Types

event_name string

Arguments

-start

To start the scope of a custom user defined event.

-end

To end the scope of a custom user defined event.

-name

User defined name of the event which encapsulates the segment.

Description

The *define_flow_summary_segment* command is used to create an artificial scope in PrimeTime's flow summary report encapsulating a set of commands and/or scripts within it. This command can be used to create an indented scope containing the set of commands and/or scripts which are invoked within the *define_flow_summary_segment -start* and *define_flow_summary_segment -end* commands.

This could be used to separate and identify a set of user scripts or PrimeTime commands within a flow summary report.

Examples

The following gives an example of define_flow_summary_segment.

```
pt_shell>set sh_flow_summary_file stdout
pt_shell> define_flow_summary_segment -start -name new_marker
pt_shell> ....   # Usual PT script
pt_shell> define_flow_summary_segment -start -name new_marker
| CUSTOM_SEGMENT_new_marker {
| | report_timing: 10.0s
| | uncaptured: 0.7s
| } 10.7s
```

Feedback

See Also

- sh_flow_summary_file

---

define_name_maps

Defines object name mapping for an application and design.

Syntax

status *define_name_maps*

```
-application application_name
-design_name design_name
-columns column_list
entries
```

Data Types

```
application_name    string
column_list         list
design_name         string
entries             list
```

Arguments

`-application application_name`

Specifies the targeting application name. Currently, only *golden_upf* is supported. The *-application* and *-design_name* options jointly specify a unique in-memory name map instance for the current session.

`-design_name design_name`

Specifies the name of the design. The *-application* and *-design_name* options jointly specify a unique in-memory name map instance for the current session.

`-columns column_list`

Specifies the column names of the name map for the *application_name*.

`entries`

Specifies name mapping entries conforming to the definition of *column_list*.

Description

The *define_name_maps* command specifies a list of object name mapping entries for a specified application. Usually, a name map is associated with a particular design name. *application_name* is predefined by the tool.

The object name map could help in certain constraint applications where the default object query or even the rule-based query algorithms could not succeed in finding the correct

objects. The typical case is when trying to apply RTL constraints onto the postoptimization netlist.

Note: It is the specific application that defines the content format and the usage model of its name map. For details about the functionality of a name map, see the documentation for the specific application.

APPLICATION golden_upf

golden_upf is the application_name assigned to golden UPF flow. The *define_name_maps* command for the *golden_upf* application typically appear in a golden UPF name mapping file, although other command such as *set_query_rules* could also exist in the golden UPF name mapping file. A *golden_upf* name map has four columns: *class*, *pattern*, *options* and *names*.

The *class*, *pattern* and *options* columns are used jointly to locate the entry in the map; the *names* column is the mapping result.

The following example illustrates the map contents and columns.

```
prompt> define_name_maps                          \\
    -application golden_upf                       \\
    -design TOP                                   \\
    -columns {class pattern options names}        \\
    {cell mid/inst/U3     {} {mid_inst/U4 mid_inst/U5}} \\
    {cell mid/inst/bot+U5 {} {mid_inst/bot/U9}}
```

The preceding name map is for the *golden_upf* application and the TOP design. The following four columns are predefined for the application:

- *class* specifies an object class name such as cell or port. The supported class names are: *cell*, *port*, *pin*, *net*, *power_domain*, *power_switch*, *supply_net*, *supply_port*, and *supply_set*.
- *pattern* specifies object matching pattern string. This is one of the object name patterns from the golden UPF file. The optional delimiter character '+' is used to indicate that a pattern is composed of a matching string (for example, "U5") and a scope (for example, scope "mid/inst/bot") for that matching string.
- *options* specifies a list of matching options for the map entry. It is derived from the golden UPF file. The supported *options* is a list of the following elements {*nocase transitive leaf*|*noleaf in*|*out*|*inout*}. *nocase* option specifies that the string in the *pattern* column should be treated case-insensitively during name lookups. For the meanings of other options, see the man page of the *find_objects* command. *leaf* and *noleaf* are mutually exclusive. *in*, *out*, and *inout* are mutually exclusive. In the preceding example, no option is used.
- *names* is a list object names where every name element is a full-path name relative to the top scope of the specified design. The *names* column is the name mapping target.

Feedback

The preceding name map would result in the following query behavior under the *golden_upf* application, although in real golden UPF flow, the queries are executed through *load_upf* command:

```
prompt> find_objects . -pattern mid/inst/U3 -object_type inst
{mid_inst/U4 mid_inst/U5}

prompt> find_objects mid/inst/bot -pattern U5 -object_type inst
{mid_inst/bot/U9}
```

See Also

- find_objects
- load_upf
- load_constraints
- set_query_rules

---

define_power_model

Defines the power model architecture.

Syntax

string *define_power_model*

```
[model_name]
[-for model_list]
```

Data Types

model_list	list
model_name	string

Arguments

`-for model_list`

PrimeTime reads and ignore this option. Complete support will be available in future releases.

`model_name string`

PrimeTime reads and ignore this option. Complete support will be available in future releases.

Description

PrimeTime reads and ignore this command. Complete support will be available in future releases.

Feedback

Examples

The following command creates ia power model named upf_model for cellA.

```
prompt> define_power_model upf_model -for cellA {
create_power_domain PD1 -elements {.} -supply {ssh1} -supply {ssh2}
#other commands ...
}
1
```

See Also

- apply_power_model

---

define_proc_attributes

Defines attributes of a Tcl procedure, including an information string for help, a command group, a set of argument descriptions for help, and so on. The command returns the empty string.

Syntax

string *define_proc_attributes*

```
proc_name
[-info info_text]
[-define_args arg_defs]
[-define_arg_groups group_defs]
[-command_group group_name]
[-return type_name]
[-hide_body]
[-hidden]
[-dont_abbrev]
[-permanent]
```

Data Types

```
proc_name       string
info_text       string
arg_defs        list
group_defs      list
group_name      string
type_name       string
```

Arguments

proc_name

Specifies the name of the existing procedure.

`-info` *`info_text`*

Provides a help string for the procedure. This is printed by the *help* command when you request help for the procedure. If you do not specify *info_text*, the default is "Procedure".

`-define_args` *`arg_defs`*

Defines each possible procedure argument for use with *help -verbose*. This is a list of lists where each list element defines one argument.

`-define_arg_groups` *`group_defs`*

Defines argument checking groups. These groups are checked for you in parse_proc_arguments. each list element defines one group

`-command_group` *`group_name`*

Defines the command group for the procedure. By default, procedures are placed in the "Procedures" command group.

`-return` *`type_name`*

Specifies the type of value returned by this proc. Any value may be specified. Some applications use this information for automatically generated dialogs etc. Please see the type_name option attribute described below.

`-hide_body`

Hides the body of the procedure from *info body*.

`-hidden`

Hides the procedure from *help* and *info proc*.

`-dont_abbrev`

Specifies that the procedure can never be abbreviated. By default, procedures can be abbreviated, subject to the value of the *sh_command_abbrev_mode* variable.

`-permanent`

Defines the procedure as permanent. You cannot modify permanent procedures in any way, so use this option carefully.

`-deprecated`

Defines the procedure as deprecated i.e. it should no longer be called but is still available.

`-obsolete`

Defines the procedure as obsolete i.e. it used to exist but can no longer be called.

Feedback

Description

The *define_proc_attributes* command associates attributes with a Tcl procedure. These attributes are used to define help for the procedure, locate it in a particular command group, and protect it.

When a procedure is created with the *proc* command, it is placed in the Procedures command group. There is no help text for its arguments. You can view the body of the procedure with *info body*, and you can modify the procedure and its attributes. The *define_proc_attributes* command allows you to change these aspects of a procedure.

Note that the arguments to Tcl procedures are all named, positional arguments. They can be programmed with default values, and there can be optional arguments by using the special argument name *args*. The *define_proc_attributes* command does not relate the information that you enter for argument definitions with *-define_args* to the actual argument names. If you are describing anything other than positional arguments, it is expected that you are also using *parse_proc_arguments* to validate and extract your arguments.

The *info_text* is displayed when you use the *help* command on the procedure.

Use *-define_args* to define help text and constraints for individual arguments. This makes the help text for the procedure look like the help for an application command. The value for *-define_args* is a list of lists. Each element has the following format:

```
arg_name option_help value_help data_type attributes
```

The elements specify the following information:

- *arg_name* is the name of the argument.
- *option_help* is a short description of the argument.
- *value_help* is the argument name for positional arguments, or a one word description for dash options. It has no meaning for a Boolean option.
- *data_type* is optional and is used for option validation. The *data_type* can be any of: string, list, boolean, int, float, or one_of_string. The default is string.
- *attributes* is optional and is used for option validation. The *attributes* is a list that can have any of the following entries:
  - "required" - This argument must be specified. This attribute is mutually exclusive with optional.
  - "optional" - Specifying this argument is optional. This attribute is mutually exclusive with "required."

- "value_help" - Indicates that the valid values for a one_of_string argument should be listed whenever argument help is shown.
- "values {<list of allowable values>}" - If the argument type is one_of_string, you must specify the "values" attribute.
- "type_name <name>" - Give a descriptive name to the type that this argument supports. Some applications may use this information to provide features for automatically generated dialogs, etc. Please see product documentation for details. This attribute is not supported on boolean options.
- "merge_duplicates" - When this option appears more than once in a command, its values are concatenated into a list of values. The default behavior is that the right-most value for the option specified is used.
- "remainder" - Specifies that any additional positional arguments should be returned in this option. This option is only valid for string option types, and by default the option is optional. You can require at least one item to be specified by also including the required option.
- "deprecated" - Specifying this option is deprecated i.e. it should no longer be used but is still available. A warning will be output if this option is specified. This attribute cannot be combined with obsolete or required.
- "obsolete" - Specifying this option is obsolete i.e. it used to exist but can no longer be used. A warning will be output if this option is specified. This attribute cannot be combined with deprecated or required.
- "min_value" <value> - Specify the minimum value for this option. This attribute is only valid for integer and float types.
- "max_value" <value> - Specify the maximum value for this option. This attribute is only valid for integer and float types.
- "default" <value> - Specify the default value for this option. This attribute is only valid for string, integer and float option types. If the user does not specify this option when invoking the command this default value will be automatically passed to the associated tcl procedure.

The default for *attributes* is "required."

Use the *-define_arg_groups* to define argument checking groups. The format of this option is a list where each element in the list defines an option group. Each element has the following format:

{<type> {<opt1> <opt2> ...} [<attributes>]}

The types of groups are

Feedback

- *"exclusive"* Only one option in an exclusive group is allowed. All the options in the group must have the same required/optional status. This group can contain any number of options.
- *"together"* If the first option in the group is specified then the second argument is also required. This type of group can contain at most two options.
- "related" These options are related to each other. An automatic dialog builder for this command may try to group these options together.

The supported attributes are:

- *{"label" <text>}* An optional label text to identify this group. The label may be used by an application to automatically build a grouping in a generated dialog.
- *"bidirectional"* This is only valid for a together group. It means that both options must be specified together.

Change the command group of the procedure using the *-command_group* command. Protect the contents of the procedure from being viewed by using *-hide_body*. Prevent further modifications to the procedure by using *-permanent*. Prevent abbreviation of the procedure by using *-dont_abbrev*.

Examples

The following procedure adds two numbers together and returns the sum. For demonstration purposes, unused arguments are defined.

```
prompt> proc plus {a b} { return [expr $a + $b]}

prompt> define_proc_attributes plus -info "Add two numbers" \\
? -define_args {
  {a "first addend" a string required}
  {b "second addend" b string required}
  {"-verbose" "issue a message" "" boolean optional}}

prompt> help -verbose plus
Usage: plus    # Add two numbers
    [-verbose]          (issue a message)
    a                   (first addend)
    b                   (second addend)

prompt> plus 5 6
11
```

In the following example, the argHandler procedure accepts an optional argument of each type supported by *define_proc_attributes*, then displays the options and values received. Note the only one of -Int, -Float, or -Bool may be specified to the command:

```
proc argHandler {args} {
  parse_proc_arguments -args $args results
```

Feedback

```
  foreach argname [array names results] {
    echo $argname = $results($argname)
  }
}

define_proc_attributes argHandler \\
  -info "Arguments processor" \\
  -define_args {
     {-Oos "oos help"       AnOos   one_of_string
        {required value_help {values {a b}}}}
     {-Int "int help"       AnInt   int     optional}
     {-Float "float help"   AFloat  float   optional}
     {-Bool "bool help"     ""      boolean optional}
     {-String "string help" AString string  optional}
     {-List "list help"     AList   list    optional}
     {-IDup "int dup help"  AIDup   int     {optional merge_duplicates}}
   } \\
  -define_arg_groups {
     {exclusive {-Int -Float -Bool}}
   }
```

See Also

- help
- parse_proc_arguments
- sh_command_abbrev_mode

---

define_qtm_attribute

Defines a new user-defined attribute for a class of Quick Timing Model (QTM) objects.

Syntax

string *define_qtm_attribute*

```
-type data_type
-class obj_class
attr_name
```

Data Types

```
data_type           string
obj_class           string
attr_name           string
```

Feedback

Arguments

`-type` *`data_type`*

Specifies the data type of the attribute. The supported data types are string, int, float, and Boolean.

`-class` *`obj_class`*

Specifies the class of QTM objects the new attribute is defined for. The valid object classes are lib, lib_cell or lib_pin. Attribute for 'lib' class applies to the library. Attribute for 'lib_cell' class applies to the QTM cell. Attribute for 'lib_pin' class applies to the QTM ports/pins.

`attr_name`

Specifies the name of the attribute.

Description

Use *define_qtm_attribute* command to define a new user-defined attribute that is applicable to QTM objects. A user-defined attribute is any attribute which PrimeTime does not understand by default. Those attributes already understood or reserved by PrimeTime for various classes of objects are considered application attributes and should not be re-defined. If you need to set the application attributes that are applicable to the classes of QTM objects, just use *set_qtm_attribute* directly. There is no need to define the attribute before setting them. But for user attributes that are not reserved by the application, you have to define them before set them on objects. Note that the *list_attributes* command will not show any attributes defined for QTM.

After save the QTM in proper library, those user-defined QTM attributes can be imported from db files. But you must define those user attributes you want to import with the *define_user_attribute* command and the *-import* option before the library with the QTM cell is read.

For more information, see the *define_user_attribute* man page.

Examples

The following example defines QTM attributes of various types.

```
pt_shell> define_qtm_attribute my_str_attr \\
           -class lib_pin -type string
pt_shell> define_qtm_attribute my_int_attr \\
           -class lib_cell -type int
pt_shell> define_qtm_attribute my_float_attr \\
           -class lib -type float
```

Feedback

See Also

- set_qtm_attribute
- remove_qtm_attribute
- define_user_attribute

---

define_scaling_lib_group

Defines a group of libraries for voltage and temperature scaling.

Syntax

string *define_scaling_lib_group*

```
[-exact_match_only]
[-best_match]
[-excluded_rail_names rail_name_list]
[-p_abstol process_threshold]
[-v_abstol voltage_threshold]
[-t_abstol temperature_threshold]
library_list
[-config_file DBFile_name]
```

Data Types

```
library_list     list
rail_name_list   list
config_file      string
```

Arguments

`-exact_match_only`

Specifies that this scaling group is analyzed in exact-match only mode. The tool skips scaling and tries to find the exact-match library. If not found, the linked library is used, and the SLG-216 warning is issued.

`-best_match`

Specifies that this scaling group is analyzed in best-match mode. The tool uses the closest (best-match) library, considering only libraries whose P, V, and T values match within given threshold values.

When no such library is found, the fallback library behavior is determined by the *timing_use_link_library_on_best_match_failure* variable.

After best-match chooses a library for a cell (whether within the thresholds or as a fallback), that library's voltage is used as the cell voltage for delay calculation.

`-excluded_rail_names` *`rail_name_list`*

Specifies a list of library voltage rails to be ignored by voltage scaling. These rails are ignored when validating scaling group formations and when performing the scaling calculations themselves, thus reducing N-dimensional multi-rail scaling to a lower dimension.

This option can be used to remove problematic rails that prevent valid formations in true scaling flows. It can also be used to remove problematic rails that do not match in exact-match flows.

This option affects all cells in the scaling group libraries. To exclude rails or *pg_pins* for specific library cells, use the *set_disable_pg_pins* command instead.

`-p_abstol` *`process_threshold`*

Specifies the best-match process threshold, in library process units. The default is 0.1.

`-v_abstol` *`voltage_threshold`*

Specifies the best-match voltage threshold, in main library voltage units. The default is 0.1.

`-t_abstol` *`temperature_threshold`*

Specifies the best-match temperature threshold, in degrees. The default is 20.

`library_list`

Specifies a list of libraries for voltage and temperature scaling. Only one of the libraries in the *library_list* option should be in the *link_path* variable; the remaining libraries are read automatically after the design is linked to complete the group.

A minimum of two libraries is required for one-dimensional scaling (voltage or temperature), and a minimum of four libraries is required for two-dimensional scaling (voltage and temperature).

`-config_file` *`DBFile_name`*

Specifies reading data from DBFile, mutually exclusive with *library_list*.

Description

The *define_scaling_lib_group* command defines a group of libraries that PrimeTime interpolates between for voltage and temperature scaling. Scaling with library groups supports timing, noise, and power analysis using Composite Current Source (CCS) model data, nonlinear delay model (NLDM) data, or a mix of both.

Normally, the *define_scaling_lib_group* command is run after the design is linked. However, it can be run before link to account for extra timing arcs in scaling libraries. For details, see "Mismatched Timing Arcs Across Scaling Group Libraries" below.

You can have more than one group to cover different portions of a design, but the groups cannot share libraries.

When a group is in effect, you cannot use the *set_rail_voltage* or *set_operating_conditions* command outside the range of the applicable group.

To see if a group is used for delay calculation, use the *report_delay_calculation* command. The report shows the results using the link library, then the scaling library group results if applicable and valid.

To report all scaling groups defined by the *define_scaling_lib_group* command, use the *report_lib_groups* command.

Mismatched Timing Arcs Across Scaling Group Libraries

The *define_scaling_lib_group* command supports mismatched timing arcs across scaling group libraries, but only in exact-match scaling group flows.

If the link library contains timing arcs that are not in one or more scaling libraries, the *define_scaling_lib_group* command can be run before or after the design is linked.

If a scaling library contains timing arcs that are not in the link library, then the *define_scaling_lib_group* command must be run before the design is linked. For example,

```
# define link library
set_app_var link_path {* lib_1.05V.db}

# define scaling library group
define_scaling_lib_group \
  -exact_match \
  {lib_0.9V.db lib_1.05V.db lib_1.3V.db}

# read and link design
read_verilog TOP.v
link_design TOP
```

This command ordering allows the design link process to be aware of any additional arcs in the scaling libraries.

Mismatched timing arc support is limited to exact-match scaling flows (voltage conditions exactly match library conditions). True scaling flows (with interpolation between libraries) still require consistent libraries within each group.

Feedback

Examples

The following example defines a scaling library group for two voltage domains about the nominal:

```
pt_shell> set link_path "* 1.1.db"
pt_shell> link_design
pt_shell> define_scaling_lib_group { 0.9.db 1.1.db 1.3.db }
```

See Also

- link_design
- report_delay_calculation
- report_lib_groups
- set_operating_conditions
- set_rail_voltage
- set_disable_pg_pins
- timing_use_link_library_on_best_match_failure

---

define_sensitivity_lib_mapping

Defines the mapping between base library and sensitivity augmented library.

Syntax

string *define_sensitivity_lib_mapping*

```
-side_file side_file
library
```

Data Types

```
side_file       string
library         collection
```

Arguments

`-side_file side_file`

Specifies the sensitivity augmented library as side file of base library.

library

Specifies the base library to define the side file.

Feedback

Description

The *define_sensitivity_lib_mapping* command defines mapping between base library and sensitivity augmented library.

Examples

The following script enables SPICE2Design flow, defines mapping between base library and sensitivity side-file, then report the mapping.

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_spice2design_analysis true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]

pt_shell> report_sensitivity_lib_mapping orig
  Lib             side_file
================================================
  orig            side.db
================================================
1
```

See Also

- report_sensitivity_lib_mapping
- reset_sensitivity_lib_mapping
- ps_enable_spice2design_analysis
- ps_enable_pvt_explorer_analysis

---

define_user_attribute

Defines a new user-defined attribute.

Syntax

string *define_user_attribute*

```
-type string | int | float | double | boolean
-classes class_list
[-range_min min]
[-range_max max]
[-one_of values]
[-import]
[-export]
[-quiet]
attr_name
```

Feedback

Data Types

class_list	list
min	double
max	double
values	list
attr_name	string

Arguments

`-type string | int | float | double | boolean`

Specifies the data type of the attribute.

`-classes` *class_list*

Defines the attribute for one or more of the classes. The valid object classes are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, *lib_pin*, *timing_path*, *clock*, or *lib_timing_arc*.

`-range_min` *min*

Specifies *min* value for numeric ranges. This is only valid when the data type is *int* or *double*. Specifying a minimum constraint without a maximum constraint creates an attribute which accepts a value >= *min*.

`-range_max` *max*

Specifies *max* value for numeric ranges. This is only valid when the data type is *int* or *double*. Specifying a maximum constraint without a minimum constraint creates an attribute which accepts a value <= *max*.

`-one_of` *values*

Provides a list of allowable strings. This is only valid when the data type is *string*.

`-import`

Imports this attribute from a design or library database.

`-export`

Exports this attribute to extract_model design or library database.

`-quiet`

Does not report any messages.

attr_name

Specifies the name of the attribute.

Feedback

Description

Use the *define_user_attribute* command to define a new user-defined attribute. A user-defined attribute is any attribute which PrimeTime does not understand by default. The *list_attributes* command shows all attributes which PrimeTime understands.

You can apply attributes to most object classes in PrimeTime. You can use these to mark interesting cells or nets, or store values you computed, and so on. PrimeTime cannot use these attributes, but you can use them in scripts, procedures, and so on. You can list the attributes you defined using the *list_attributes* command.

User-defined attributes can be imported from a design or library database after they have been defined. A design or library database is a db or ddc format file, or a Milkyway database. You must define attributes you want to import with the *-import* option. Note that imported attributes do not appear on the design objects until the design is linked. Attributes imported from a design or library database are inherited through the hierarchy. Note that attributes defined for designs are inherited onto cells, and attributes defined for ports are inherited onto pins. The *define_user_attribute* command sets up these relationships for you automatically if you do not do it explicitly.

Examples

The following example defines attributes of various types. Note that more than one class can be specified. Also note that attr_i is defined as imported from a design or library database. Finally, attribute design1 is imported from a design or library database, and PrimeTime defines it for cells as well.

```
pt_shell> define_user_attribute attr_s \\
            -classes {cell net} -type string
pt_shell> define_user_attribute attr_i \\
            -classes cell -type int -import
pt_shell> define_user_attribute design1 \\
            -classes design -type int -import
Information: Inferred definition of attribute 'design1' for class 'cell'
        because it is imported for class 'design' (ATTR-7)
```

In the following example, attr_ir1 is defined as >= 0 and <= 100, attr_ir2 is defined as >= 0 with no maximum, and attr_ir3 is defined as <= 100 with no minimum.

```
pt_shell> define_user_attribute attr_ir1 \\
            -classes cell -type int \\
            -range_min 0 -range_max 100
pt_shell> define_user_attribute attr_ir2 \\
            -classes cell -type int -range_min 0
pt_shell> define_user_attribute attr_ir3 \\
            -classes cell -type int -range_max 100
```

In the following example, define attribute attr_oo for cells. The attribute is a string, but can only be set to A, B, C, or D.

Feedback

```
pt_shell> define_user_attribute attr_oo \\
            -classes cell -type string -one_of {A B C D}
```

After these attributes have been defined, you can list the attribute definitions using the *list_attribute* command. Notice how the 'attr_i' attribute is marked as importable from db, meaning a design or library database.

```
pt_shell> list_attributes

****************************************
Report : List of Attribute Definitions
 ...
****************************************

Properties:
    A - Application-defined
    U - User-defined
    I - Importable from db (for user-defined)

Attribute Name   Object  Type      Properties  Constraints
-------------------------------------------------------------
attr_i           cell    int       U,I
attr_ir1         cell    int       U           0 to 100
attr_ir2         cell    int       U           >= 0
attr_ir3         cell    int       U           <= 100
attr_oo          cell    string    U           A, B, C, D
attr_s           cell    string    U
attr_s           net     string    U
design1          design  int       U,I
design1          cell    int       U
```

In the following example, attr_xyz is defined and exported as string for design and port, and set values "abc" and "def" on designX and portX respectively.

```
pt_shell> define_user_attribute attr_xyz \\
            -classes {design port} -type string \\
            -export
pt_shell> set_user_attribute [get_design designX] attr_xyz abc
pt_shell> set_user_attribute [get_port portX] attr_xyz def
```

After extract_model, the attribute attr_xyz and its values are expected in .lib.

```
pt_shell> extract_model -output mymodel -format {db lib}
Model extraction : Clock period/min_pulse_width arcs extracted at context
 slew.
Wrote the LIB file ./ETM/mymodel/mod.lib
Wrote model to './ETM/mymodel/mod_lib.db'
 ...
pt_shell>sh cat ./ETM/mymodel/mod.lib
 ...
cell( designX ) {
  ...
```

```
  attr_xyz : abc;
  ...
 ...
pin("portX") {
        ...
        attr_xyz : def;
        ...
} /* end of pin portX */
 ...
} /* end of cell */
 ...
```

See Also

- get_attribute
- list_attributes
- read_db
- read_ddc
- read_milkyway
- remove_user_attribute
- report_attribute
- set_user_attribute
- extract_model

---

derive_clocks

Creates clocks on source pins in design.

Syntax

string *derive_clocks*

```
-period period_value
[-waveform edge_list]
```

Data Types

```
period_value      float
edge_list         list
```

Feedback

Arguments

`-period` *`period_value`*

Specifies the clock period of the automatically derived clocks. The clock period has a value greater than or equal to zero (value >= 0).

`-waveform` *`edge_list`*

Specifies the rise and fall edge times of the clock, in library time units, over an entire clock period. It defines the clock edge specification. The first time that is listed is a rising transition; typically the first rising transition after time zero. There must be an even number of increasing times and alternating rise and fall times. If you do not specify an *edge_list* value, the command assumes a default waveform that has a rise edge of *0.0* and a fall edge of *period_value*/2.

Description

Creates clocks on source pins in design. This command finds the clock source ports and pins that must have clocks defined so that every register clock pin has a clock. Then on each source set, the command creates a clock in the same way as the *create_clock* command does. The clocks are created with the specified waveform or period. For a description of commands that use clock objects, see the *create_clock* command man page.

Examples

The following example creates a clock on the source pin with a clock period of 20:

```
pt_shell> derive_clocks -period 20
```

See Also

- create_clock
- get_clocks
- group_path
- remove_clock
- report_clock
- set_clock_latency
- set_clock_uncertainty
- set_input_delay
- set_output_delay

Feedback

disconnect_net

Disconnects a net from specified pins or ports, or from all pins and ports.

Syntax

```
status disconnect_net
  -all
  net
  object_spec
```

Data Types

```
net           string
object_spec   list
```

Arguments

`-all`

Indicates that all pins and ports are to be disconnected from *net*. The *-all* and *object_spec* options are mutually exclusive.

net

Specifies the name of the net to be disconnected.

object_spec

Specifies a list of pins or ports to be disconnected from *net*. The *-all* and *object_spec* options are mutually exclusive.

Description

The *disconnect_net* command disconnects pins and ports from a *net* in the current design. Like all other netlist editing commands, for the *disconnect_net* command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. the *disconnect_net* command returns a 1 if successful and a 0 if unsuccessful.

The *net* option and each pin and port specified in the *object_spec* option must be in scope; that is, at or below the current instance. There are three rules for *disconnect_net*. They are:

- You cannot disconnect an unconnected pin or a port.
- You cannot disconnect a pin or a port that is not connected to *net*.
- You cannot disconnect a hierarchical pin or a port that has the same name as *net*.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported

using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is re-linked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Examples

In the following example, the first command attempts to connect port in1 to net new_net1, but in1 is already connected to a net. The second command determines the net to which in1 is connected; the third command disconnects the port from its net. The fourth and final command connects in1 to new_net1.

Feedback

```
pt_shell> connect_net new_net1 [get_ports in1]
Error: Could not connect 'in1' to 'new_net1'
        Object is already connected - disconnect it first (NED-042)
Error: No changes made. (NED-040)
0
pt_shell> set pnet [get_nets -of_objects [get_ports in1]]
{"in1"}
pt_shell> disconnect_net $pnet [get_ports in1]
Information: Disconnected 'in1' from 'net1'. (NED-019)
1
pt_shell> connect_net new_net1 [get_ports in1]
Information: Connected 'in1' to 'new_net1'. (NED-018)
1
```

See Also

- connect_net
- create_cell
- create_net
- size_cell
- swap_cell
- write_changes

---

drive_of

Determines the drive resistance of the specified library cell pin. The *drive_of* command is a DC Emulation command provided for compatibility with Design Compiler.

Syntax

float *drive_of*

```
[-rise]
[-fall]
[-wire_drive]
[-piece val]
lib_cell_pin
```

Data Types

val	string
lib_cell_pin	string

Feedback

Arguments

`-rise`

Gets rise drive value.

`-fall`

Gets fall drive value.

`-wire_drive`

Not supported in PrimeTime.

`-piece` *`val`*

Not supported in PrimeTime.

`lib_cell_pin`

Specifies the name of the library cell pin whose drive value is to be returned. This value can be either the drive resistance or a collection that contains the library cell pin.

Description

The *drive_of* command returns the drive resistance of the given library cell pin. If neither the *-rise* nor *-fall* option is specified, the greater value of rise and fall is returned.

The *drive_of* command exists in PrimeTime for compatibility with Design Compiler. Complete information about the *drive_of* command can be found in the Design Compiler documentation. The supported method for getting the capacitance of a library cell pin is by using the *get_attribute* command with either the *drive_resistance_rise* or *drive_resistance_fall* attribute.

See Also

- get_attribute

---

e

---

echo

Echos arguments to standard output.

Syntax

string *echo*

```
[-n]
[arguments]
```

Feedback

Data Types

arguments string

Arguments

-n

Suppresses the new line. By default, *echo* adds a new line.

arguments

Specifies the arguments to be printed.

Description

The *echo* command prints out the value of the given arguments. Each of the arguments are separated by a space. The line is normally terminated with a new line, but if the *-n* option is specified, multiple *echo* command outputs are printed onto the same output line.

The *echo* command is used to print out the value of variables and expressions in addition to text strings. The output from *echo* can be redirected using the > and >> operators.

Examples

The following are examples of using the *echo* command:

```
prompt> echo
"Running version" $sh_product_version
Running version v3.0a

prompt> echo -n "Printing to" [expr (3 - 2)] > foo
prompt> echo " line." >> foo
prompt> sh cat foo
Printing to 1 line.
```

See Also

- sh

---

enable_primetime_icc_consistency_settings

Sets all recommended variables with values for consistency with IC Compiler.

Syntax

string *enable_primetime_icc_consistency_settings*

[-all]

Feedback

Arguments

`-all`

Applies the recommended values to all variables, including flow-specific variables. If you do not use this option, flow-specific variables are not adjusted or reported.

Description

The *enable_primetime_icc_consistency_settings* command sets all common variables to have values that are recommended for consistency. By default, the flow-specific and feature gap variables are ignored.

Examples

```
pt_shell> enable_primetime_icc_consistency_settings
set rc_degrade_min_slew_when_rd_less_than_rnet false
# rc_degrade_min_slew_when_rd_less_than_rnet changed to true
set timing_clock_gating_propagate_enable true
# timing_clock_gating_propagate_enable changed to false
```

---

error_info

Prints extended information on errors from the last command.

Syntax

```
string error_info
```

Arguments

This *error_info* command has no arguments.

Description

The *error_info* command is used to display information after an error has occurred. Tcl collects information showing the call stack of commands and procedures. When an error occurs, the *error_info* command can help you to focus on the exact line in a block that caused the error.

Examples

This example shows how *error_info* can be used to trace an error. The error is that the iterator variable "s" is not dereferenced in the 'if' statement. It should be '$s == "a" '.

```
prompt> foreach s $my_list {
?         if { s == "a" } {
?           echo "Found 'a'!"
?         }
?       }
```

Feedback

```
Error: syntax error in expression " s == "a" "
       Use error_info for more info. (CMD-013)
shell> error_info
Extended error info:
syntax error in expression " s == "a" "
    while executing
"if { s == a } {
     echo "Found 'a'"
}"
    ("foreach" body line 2)
    invoked from within
"foreach s [list a b c] {
   if { s == a } {
     echo "Found 'a'"
}
}"
 -- End Extended Error Info
```

---

estimate_clock_gate_max_power

Find the worst-case internal energy when-condition in the liberty model for memory cells and set those for each of instances

Syntax

```
int estimate_clock_gate_max_power
[-target_ts]
```

Data Types

instance_list	list

Arguments

`-target_ts`

To set the target average toggle savings given by the user.

Description

Find the worst-case internal energy when-condition in the liberty model and set those for each of instances Ignore all propagated TR and SP values to the nets which are part of the when-condition If there is logic conflict with setting it will be ignored. For all other pins which are not part of the selected when-condition use the propagated TR and SP.

- PrimePower will use user defined conditions for power calculation for specific instances • Power calculation – Maximum of weighted sum of states that match the condition specified by user. – No change to toggle rate propagation • User constraint form (UI preliminary subject to change) – Set inst_power_condition <inst_name> {Pin_name bool_value derate_value,…} <related_pin> – inst_name and related_pin is mandatory – Ex: Set inst_power_condition top/blockA/mem1 {RCS 1 0.5, WCS 1 0.5} CLK – User

does not need to specify conditions on all the other pins that are not included in the inst_power_condition command – Sum of derate values in an inst_power_condition cannot exceed 1.0 – If multiple inst_power_conditions defined for an instance, then tool will pick last statement one for power analysis for the same inst_name and related_pin

Example1: Single port memory – estimate_clock_gate_max_power top/A/InstSp1 {“CS & !WE & !PD & SLEEPB” 1 0.5, “CS & WE & !PD & SLEEPB” 1 0.0} CK – Matching states – S2 = “CS & !WE & !PD & SLEEPB” 1 0.5 with CK S2_derate = 0.5 – S1 = “CS & WE & !PD & SLEEPB” 1 0.0 with CK S1_derate = 0.0 – total_derate_matching_states = 0.5 + 0.0 = 0.5 – Non_matching states – S3 = !CS & !PD & SLEEPB – total_derate_non_matching_states = 1.0 – total_derate_matching_states = 1-0.5=0.5 – Power calculation for supply VCC – matching_state_pwr – S2_pwr*S2_derate + S1_pwr*S1_derate = 40.0*0.5 + 50.0*0.0 = 20 – Non_matching_state_pwr – S3_pwr * total_derate_non_matching_states = 3.0*0.5 = 1.5 – Total_pwr_vcc = matching_state_pwr + non_matching_state_pwr = 20 + 1.5 = 21.5 – Similar calculation will be performed for cvcc – total_power = total_pwr_vcc + total_power_cvcc

This command should only be run in average mode. update_power should be used to calculate power and report_power shows the maximum power values used for memory cells.

The *state_condition* option can be used to change the when condition of the state used for derating.

The *target_ts* option can be used to set the derate value to be used for a given state condition.

The *pin* option can be used to specify the input pin which needs to be used for derating and power calculation.

Examples

The following example shows how to set maximum memory power on a single cell in the design. The command dumps out the output in the report_power command.

```
pwr_shell> estimate_clock_gate_max_power I_SINGLE_PORT -target_ts 0.5
estimate_clock_gate_max_power I_SINGLE_PORT -target_ts 0.5

1
```

The following example sets the derate value of 0.5 for cell "I_TCAM" and creates an output file "debug_s1_report.rpt"

```
pwr_shell> estimate_clock_gate_max_power I_TCAM -target_ts 0.5

estimate_clock_gate_max_power I_TCAM -target_ts 0.5
1
```

Feedback

See Also

- update_power
- report_power

---

estimate_clock_network_power

Virtually generate a clock tree and estimate its power.

Syntax

string *estimate_clock_network_power*

```
lib_cell
[-max_fanout fanout]
[-wire_load_model wire_load_name]
[-library lib_name]
[-input_transition transition]
[-clocks clock_list]
[-include_registers]
[-ignore_clock_gating]
```

Data Types

```
lib_cell              string
fanout                int
wire_load_name        string
lib_name              string
transition            float
clock_list            list
```

Arguments

lib_cell

Specifies the buffer or inverter to be used to build the clock tree.

-max_fanout *fanout*

Specifies the maximum fanout number that the specified buffer can drive. The default number is 32.

-wire_load_model *wire_load_name*

Specifies the wire load model to be used to compute the wire capacitance for the nets in the clock tree. The default is the wire load model set to the design.

-library *lib_name*

Specifies the library where the specified wire load model is defined.

`-input_transition` *`transition`*

Specifies the clock input transition. If not specified, use the clock transition annotated to the clock by the *set_clock_transition* command. If there is no annotation either, the default to 0.

`-clocks` *`clock_list`*

Specifies the clocks to be estimated for clock tree power. The default is all the clocks in the design.

`-include_registers`

Indicates the power of registers driven by the clock is also included in the power report.

`-ignore_clock_gating`

Indicates that existing clock tree cells like the clock gating cells are ignored when building the clock tree.

Description

The *estimate_clock_network_power* command virtually creates a clock tree for each clock and calculates power for the generated clock tree. The clock trees are built on the fly and the original design is not touched or modified. The following describes how the clock tree is built:

Find all the nets in the clock network. For each net in the clock network, insert buffers to build a clock tree, so that the fanout of each buffer in the clock tree is no bigger than the max fanout. The command tries to build a tree as balanced as possible such that the delays from the root to the leafs do not vary a lot. All the driven registers are put at the same and lowest level of the tree. The deviation of buffer fanouts at the same tree level is kept as small as possible.

The output load of each buffer is calculated using wire load model. The input transition of each buffer propagates from the root of the tree. The switching activity for each inserted buffer is equal to clock frequency or from annotation (SAIF or set_switching_activity).

If there are clock gating cells in the original clock network, the command builds a separate tree based on the fanout of the clock gating cell using the same rule for the clock tree, and this separate tree is considered as a branch of the whole clock tree, and also balances with other branches. Then the clock gating effect is accounted for by the annotated or, if not annotated, propagated switching activity. In order for you to investigate savings from clock gating, the command has the *-ignore_clock_gating* option to estimate the clock tree power by ignoring the clock gating cells and compare with the power that considers the clock gating effect. The power consumed by the registers driven by the clock is also included in the report if the *-include_registers* option is specified.

Feedback

Examples

In the following example, clock tree power is estimated for clock CLK by building the clock tree using the buffer buf1a1 of library ssc_core_typ and wire load model in the design is used to calculate the wire capacitance. The clock input transition is 0.2.

```
pt_shell> create_clock -name CLK -period 10 clk
pt_shell> set_clock_transition 0.2 CLK
pt_shell> estimate_clock_network_power ssc_core_typ/buf1a1
```

See Also

- report_power

---

estimate_eco

Estimates delay changes for the *size_cell* and *insert_buffer* commands.

Syntax

string *estimate_eco*

```
[-lib_cells lib_cell]
[-verbose]
[-max]
[-min]
[-rise]
[-fall]
[-from from_pin]
[-rise_from from_pin]
[-fall_from from_pin]
[-through_arc timing_arc]
[-to   to_pin]
[-rise_to   to_pin]
[-fall_to   to_pin]
[-data_pin]
[-inverter_pair]
[-current_library]
[-libraries lib_spec]
[-significant_digits digits]
[-nosplit]
[-type size_cell | insert_buffer]
[-power]
[-output_format_in_list]
[-sort_by sort_type]
object_list
```

Data Types

```
lib_cell        list
from_pin        string
```

```
to_pin          string
lib_spec        string
digits          integer
sort_type       string
object_list     list
```

Arguments

`-lib_cells` *lib_cell*

Specifies the list of library cells that can be used for ECO estimation. Specifying this list is mandatory for buffer insertion and optional for cell sizing.

`-verbose`

Generates a very detailed estimation report for each candidate library cell.

`-max`

Specifies only maximum delay calculation reporting. This is the default.

`-min`

Specifies only minimum delay calculation reporting.

`-rise`

Restricts reporting to rising transitions at the stage driver output pin. The default is to consider both rising and falling transitions.

`-fall`

Restricts reporting to falling transitions at the stage driver output pin.

`-from` *from_pin*

Specifies the *from_pin* of the path from which to get the estimated timing. The *from_pin* must be one of the inputs of the cell being estimated. It cannot be a hierarchical pin. No more than one of *-from*, *-rise_from*, or *-fall_from* can be used in the command.

This option can't be used together with *-rise_to*, *-fall_to*.

`-rise_from` *from_pin*

Like the *-from* option, but only rising transitions at the pin are considered.

`-fall_from` *from_pin*

Like the *-from* option, but only falling transitions at the pin are considered.

`-through_arc` *timing_arc*

Specifies a dedicated timing arc from the input to the output of the "from" cell in the path specified by the *-from* and *-to* options.

The argument of the *-through_arc* option is a cell timing arc gathered by the *get_timing_arcs* command. Use this option to restrict the timing analysis to only specific types of timing arcs through the cell. See the EXAMPLES section for details.

-through_arc should be used for cell-arcs only.

`-to` *`to_pin`*

Specifies the *to_pin* of the path from which to get the estimated timing. The *to_pin* must be an input pin of a cell in the direct fanout of the cell being estimated. It cannot be a hierarchical pin. No more than one of *-to*, *-rise_to*, or *-fall_to* can be used in the command.

This option can't be used together with *-rise_from*, *-fall_from*.

`-rise_to` *`to_pin`*

Like the *-to* option, but only rising transitions at the pin are considered.

`-fall_to` *`to_pin`*

Like the *-to* option, but only falling transitions at the pin are considered.

`-data_pin`

Estimates the slack of the related data-pin of a sequential cell. The related data-pin can be the D-pin of a sequential cell, a reset/clear pin of a sequential cell, a clock-gating enable pin, or the D-pin of a nonsequential cell.

To use the *-data_pin* option, you must first add *data_pin_slack* to the *eco_estimation_output_columns* variable.

`-inverter_pair`

Specifies that an inverter pair is to be inserted when the ECO type is *insert_buffer*.

`-current_library`

Searches only the cell's current library to resolve the *lib_cell* name.

`-libraries` *`lib_spec`*

Resolves *lib_cell* using only cells from the listed libraries.

`-significant_digits` *`digits`*

Specifies the number of digits after the decimal point displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which is 2 by default. This option controls only the number of digits displayed, not the precision used internally for analysis.

`-nosplit`

Prevents line-splitting and facilitates writing software to extract information from the report output.

`-type size_cell | insert_buffer`

Specifies either *size_cell* or *insert_buffer* ECO estimation. If you do not use this option, the default is *size_cell*.

`-output_format_in_list`

By default, estimate_eco output will be flush to console. When specified, estimation_eco results will be kept inside list.

For example, originally we have estimation report like this,

```
pt_shell> estimate_eco u0 -from u1/A -to u2/Z -nosplit;
delay type : max_rise
lib cell        arrival   slack
--------------------------------
*lib1/BUFX1     1.5263    5.0714
 lib1/BUFX2     1.5305    5.0669
 lib1/BUFX3     1.5284    5.0690
```

Then, we can use option '-output_format_in_list' to keep estimation into list.

```
pt_shell> set results [estimate_eco u1 -nosplit
 -output_format_in_list];
pt_shell> set line_result [lindex $results 0];
*lib1/BUFX1 1.5263 5.0714
pt_shell> echo [lindex $line_result 1];
1.5263
pt_shell> echo [lindex $line_result 2];
5.0714
```

`-sort_by` *`sort_type`*

Specifies the method of sorting library cells reported. The following types are allowed: *area*, *stage_delay*, *arrival*, *slack*, *transition*, *max_transition*, *min_transition*, *max_capacitance*, or *min_capacitance*. If you do not use this option, the default is *area*.

`-power`

Estimates changes in power in addition to timing.

`object_list`

Specifies the name of object to analyze. For *size_cell*, the object is a cell to be sized. For *insert_buffer*, the object is a pin where the buffer is to be inserted.

Feedback

Description

The *estimate_eco* command provides a method to estimate the timing effects of certain ECO commands, without actually incurring a timing update to compute the slack. As its name implies, the results of this command are estimates only, and are expected to reasonably predict, but not necessarily match, the actual slack resulting from the ECO command.

ECO estimation is supported for two ECO commands: *size_cell* and *insert_buffer*. This is specified with the *-type* argument, which takes a value of *size_cell* or *insert_buffer*, which matches the corresponding command name. Both buffer insertion and double-inverter insertion are supported for *insert_buffer* estimation.

When performing *insert_buffer* estimation, the *-lib_cell* option is required and must supply a list of one or more buffer library cells as buffer insertion candidates. The buffer cells can be specified in the form of *lib_cell* collection objects, a list of library_name/base_name names, or simply a list of base_name library cell names. The library cell specifications follow the same rules as the *insert_buffer* command. For more information, see the *insert_buffer* man page.

When performing *size_cell* estimation, the *-lib_cell* option is optional; the default is the list of alternative library cells for the specified instance cell as obtained by the *get_alternative_lib_cells* command. When the *-lib_cell* option is specified, the candidate library cells can be specified in the form of lib_cell collection objects, a list of library_name/base_name names, or simply a list of base_name library cell names. The library cell specifications follow the same rules as the *size_cell* command. To improve runtime, manually specified candidate cells are not validated for functional equivalence before estimation is performed. For more information, see the *size_cell* man page.

For both buffer insertion and cell sizing, to limit the candidate cells to a specific library, you can use the *-current_library* and *-libraries* options of the *get_alternative_lib_cells* command, and pass the resulting lib_cell collection to the *estimate_eco* command.

The *estimate_eco* command generates a report that predicts the timing and slack effects of one or more potential library cell candidates for the *size_cell* or *insert_buffer* command. By default, the estimates are shown in a compact form that lists one candidate library cell per line, with stage delay, arrival, and slack information columns. A verbose mode is also available with the *-verbose* option, which shows a separate detailed estimation report for each candidate library cell.

For example, in the verbose mode, a stage delay is divided into a cell delay, an output net delay, a buffer delay, and a delta delay. The cell delay indicates the cell arc delay of the driver cell of the net. The output net delay is the net arc delay from the driver pin of the net to the slack-critical load pin. The buffer delay is displayed only for buffer insertion estimate and indicates the cell arc delay of the buffer cell. The delta delay indicates the crosstalk delta delay of the net.

The *-min* and *-max* options are available to specify whether minimum-delay hold timing, maximum-delay setup timing, or both should be shown. By default, the maximum-delay timing is shown. The *-rise* and *-fall* options are available to specifically request rise/fall timing. By default, both rise and fall information is shown. If only the *-rise* or *-fall* option is specified, only that information is shown.

The estimate computation takes into account the incoming transition times, the output loading, the fanout loading, the detailed parasitics (if present), and the driver characteristics of the candidate cells. The rise/fall timing shown in the report represents the worst timing through the cell output pins in the corresponding direction. In the verbose report, the arrival is the arrival time at the input pin of the load cell driven by the resized or inserted cell, and the stage delay is the delay of the inserted/resized cell and the driven net.

In the compact report without the verbose option, the reported arrival time, slack, and transition time are the estimated values at the input pin of the slack-critical load cell. Maximum transition, minimum transition, maximum capacitance, and minimum capacitance values are the worst values of the pins along the path to the slack-critical load pin. Individual values are shown in verbose reporting.

To perform slack and arrival estimation, the *timing_save_pin_arrival_and_slack* variable should be set to *true* ahead of time. If this variable is not already set to *true*, the *estimate_eco* command automatically sets it to *true* and updates the design timing before it proceeds with ECO estimation.

Estimated data can be selectively reported if the content of the *eco_estimation_output_columns* variable is changed. By default, only *area*, *stage_delay*, *arrival*, and *slack* are reported. Others, such as *max_transition* and *max_capacitance*, can be reported by changing the variable. For more information, see the *eco_estimation_output_columns* man page.

The *-power* option works only for estimation using *size_cell*. By default, the total power of a cell is reported. The verbose option shows detailed power information like dynamic and leakage power. To use this option, power analysis should be enabled ahead of time by setting the *power_enable_analysis* variable to *true*. The *-power* option triggers a power update if the design power is not already up-to-date. The power numbers displayed are in Watts.

Examples

The following example shows a minimum and maximum delay summary estimation of four potential buffer cell insertions at an instance pin:

```
pt_shell> estimate_eco -type insert_buffer -min -max \\
            -lib_cells {BUFX2 BUFX4 DLY1X1 DLY2X1} \\
            wishbone/TxDone_wb_reg/D

  delay type : max_rise
```

```
 lib cell                  area  stage_delay      arrival         slack
 ---------------------------------------------------------------------
 slow/DLY1X1              19.96        0.857        2.076        14.859
 slow/DLY2X1              19.96        1.218        2.437        14.498
 slow/BUFX4               16.63        0.639        1.858        15.077
 slow/BUFX2               13.31        0.647        1.867        15.069
*No buffer                 0.00        0.491        1.711        15.225

 delay type : max_fall
 lib cell                  area  stage_delay      arrival         slack
 ---------------------------------------------------------------------
 slow/DLY1X1              19.96        0.746        1.966        14.767
 slow/DLY2X1              19.96        1.088        2.308        14.425
 slow/BUFX4               16.63        0.514        1.734        14.999
 slow/BUFX2               13.31        0.534        1.754        14.979
*No buffer                 0.00        0.354        1.573        15.160

 delay type : min_rise
 lib cell                  area  stage_delay      arrival         slack
 ---------------------------------------------------------------------
 slow/DLY1X1              19.96        0.366        2.076         0.296
 slow/DLY2X1              19.96        0.727        2.437         0.657
 slow/BUFX4               16.63        0.148        1.858         0.077
 slow/BUFX2               13.31        0.156        1.867         0.086
*No buffer                 0.00        0.000        1.711        -0.070

 delay type : min_fall
 lib cell                  area  stage_delay      arrival         slack
 ---------------------------------------------------------------------
 slow/DLY1X1              19.96        0.679        1.966         0.098
 slow/DLY2X1              19.96        1.021        2.308         0.440
 slow/BUFX4               16.63        0.447        1.734        -0.134
 slow/BUFX2               13.31        0.467        1.754        -0.114
*No buffer                 0.00        0.286        1.573        -0.294
```

The following example shows a minimum delay verbose estimation of a single potential buffer cell insertion at an instance pin:

```
pt_shell> estimate_eco -type insert_buffer \\
            -min -verbose -lib_cells {DLY1X1} \\
            wishbone/TxDone_wb_reg/D


  cell name       : wishbone/TxDoneSync1_reg
  current lib cell: slow/DFFRHQX1

  buffer lib cell: slow/DLY1X1
  min_rise                                 current          estimate
  ---------------------------------------------------------------
  input net delay                            0.012             0.012
  stage delay                                0.000             0.366
    cell delay                               0.000             0.000
    output net delay                         0.000             0.000
```

```
    buffer delay                          0.000            0.366
    delta delay                           0.000            0.000
  arrival time                            1.711            2.076
  slack                                  -0.070            0.296

  buffer lib cell: slow/DLY1X1
  min_fall                              current          estimate
  ------------------------------------------------------------------
  input net delay                         0.011            0.011
  stage delay                             0.286            0.679
    cell delay                            0.286            0.286
    output net delay                      0.000            0.000
    buffer delay                          0.000            0.392
    delta delay                           0.000            0.000
  arrival time                            1.573            1.966
  slack                                  -0.294            0.098
```

The following example shows a maximum delay summary estimation of potential cell instance resize operations for a given inverter cell name pattern:

```
pt_shell> estimate_eco -type size_cell \\
            -lib_cells {INV*} wishbone/bd_ram/U9/U6200

  delay type : max_rise
  lib cell                 area  stage_delay      arrival        slack
  ------------------------------------------------------------------
  slow/INVX20             63.20        1.062        9.340        0.019
  slow/INVX16             56.55        1.173        9.451       -0.092
  slow/INVX12             43.24        1.347        9.624       -0.266
  slow/INVX8              19.96        1.682        9.961       -0.602
 *slow/CLKINVX4           13.31        2.727       11.005       -1.646
  slow/INVX3              13.31        3.423       11.701       -2.342
  slow/INVX4              13.31        2.770       11.048       -1.689
  slow/INVX2               9.98        4.943       13.220       -3.861
  slow/INVX1               6.65        9.269       17.546       -8.187
  slow/INVXL               6.65       13.239       21.516      -12.157

  delay type : max_fall
  lib cell                 area  stage_delay      arrival        slack
  ------------------------------------------------------------------
  slow/INVX20             63.20        0.855        7.542        1.843
  slow/INVX16             56.55        0.940        7.627        1.758
  slow/INVX12             43.24        1.074        7.761        1.624
  slow/INVX8              19.96        1.279        7.967        1.418
 *slow/CLKINVX4           13.31        2.731        9.418       -0.033
  slow/INVX3              13.31        2.278        8.965        0.420
  slow/INVX4              13.31        1.901        8.588        0.797
  slow/INVX2               9.98        3.192        9.879       -0.494
  slow/INVX1               6.65        5.420       12.107       -2.722
  slow/INVXL               6.65        7.474       14.161       -4.776
```

The following example shows a maximum delay verbose estimation of potential cell instance resize operations for specific user-specified alternative cells:

```
pt_shell> estimate_eco -type size_cell -verbose \\
            wishbone/bd_ram/U9/U6200 -lib_cells {INVX20}

  cell name       : wishbone/bd_ram/U9/U6200
  current lib cell: slow/CLKINVX4

  new lib cell: slow/INVX20
  max_rise                                  current          estimate
  ------------------------------------------------------------------
  area                                       13.310            63.200
  input net delay                             0.005             0.005
  stage delay                                 2.727             1.062
    cell delay                                2.609             0.957
    output net delay                          0.118             0.105
    delta delay                               0.000             0.000
  arrival time                               11.005             9.340
  slack                                      -1.646             0.019

  new lib cell: slow/INVX20
  max_fall                                  current          estimate
  ------------------------------------------------------------------
  area                                       13.310            63.200
  input net delay                             0.005             0.005
  stage delay                                 2.731             0.855
    cell delay                                2.612             0.751
    output net delay                          0.118             0.104
    delta delay                               0.000             0.000
  arrival time                                9.418             7.542
  slack                                      -0.033             1.843
```

The following example reports only the transition and maximum transition parameters.

```
pt_shell> set eco_estimation_output_columns "transition max_transition"
pt_shell> estimate_eco -type size_cell -max -rise ram/U9/U6200

  delay type : max_rise
  lib cell                 trans    max_trans
  ----------------------------------------
  slow/INVX20              9.340        0.019
  slow/INVX16              9.451       -0.092
  slow/INVX12              9.624       -0.266
  slow/INVX8               9.961       -0.602
 *slow/CLKINVX4           11.005       -1.646
  slow/INVX3              11.701       -2.342
  slow/INVX4              11.048       -1.689
  slow/INVX2              13.220       -3.861
  slow/INVX1              17.546       -8.187
  slow/INVXL              21.516      -12.157
```

The following example shows power estimation for various library cells.

```
pt_shell> estimate_eco -type size_cell -lib_cell [get_lib_cell
 tcbn90gthpwc/INVD*] \\
  [get_cell buf5] -max -rise -power -significant_digits 4

delay type : max_rise
  lib cell                      area stage_delay     arrival       slack
      power

 ------------------------------------------------------------------------
----------
  tcbn90gthpwc/INVD24         21.1680      0.0409      0.4230      0.2609
  2.189e-05
  tcbn90gthpwc/INVD20         18.3456      0.0420      0.4183      0.2656
  1.799e-05
  tcbn90gthpwc/INVD16         14.8176      0.0436      0.4140      0.2698
  1.424e-05
  tcbn90gthpwc/INVD12         11.2896      0.0476      0.4119      0.2717
  1.068e-05
  tcbn90gthpwc/INVD8           7.7616      0.0535      0.4122      0.2710
  7.316e-06
  tcbn90gthpwc/INVD6           6.3504      0.0591      0.4151      0.2676
  5.756e-06
  tcbn90gthpwc/INVD4           4.2336      0.0693      0.4223      0.2596
  4.238e-06
  tcbn90gthpwc/INVD3           3.5280      0.0806      0.4323      0.2491
  3.574e-06
  tcbn90gthpwc/INVD2           2.8224      0.0993      0.4495      0.2311
   2.87e-06
  tcbn90gthpwc/INVD1           2.1168      0.1464      0.4953      0.1833
  2.294e-06
 *tcbn90gthpwc/INVD0           2.1168      0.2414      0.5896      0.0866
  2.005e-06
```

The following example shows how to do estimate_eco for one buffer 'u0' with '-from' and '-to' options.

```
pt_shell> set eco_estimation_output_columns "area arrival slack ";
pt_shell> estimate_eco u0 -from u0/A -to cg1/EN -max -rise;

  delay type : max_rise
  lib cell                  area        arrival     slack
  ---------------------------------------------------------
  slow/HSBULT16_BUF_24      1.9699      0.0058      2.1939
  slow/HSBULT16_DEL_L6_8    1.3997      0.0718      2.1263
  slow/HSBULT16_BUF_S_16    1.3478      0.0055      2.1943
  slow/HSBULT16_BUF_16      1.3478      0.0055      2.1943
  slow/HSBULT16_BUF_14      1.1923      0.0055      2.1943
  slow/HSBULT16_BUF_D_16    1.1405      0.0080      2.1914
  slow/HSBULT16_BUF_12      1.0368      0.0055      2.1943
```

```
  slow/HSBULT16_BUF_S_12     1.0368      0.0054      2.1943
 *slow/HSBULT16_BUF_U_1      0.2074      0.0053      2.1942
```

The following example shows how to do estimate_eco for one buffer 'u0' with '-rise_from/-fall_fror' and '-rise_to/-fall_to' options.

```
pt_shell> set eco_estimation_output_columns "area arrival slack ";
pt_shell> estimate_eco u0 -rise_from u0/A -fall_to cg1/EN -max ;

  delay type : max_rise
  lib cell                    area        arrival     slack
  ---------------------------------------------------------
  slow/HSBULT16_BUF_24        1.9699      0.0058      2.1939
  slow/HSBULT16_DEL_L6_8      1.3997      0.0718      2.1263
  slow/HSBULT16_BUF_S_16      1.3478      0.0055      2.1943
  slow/HSBULT16_BUF_16        1.3478      0.0055      2.1943
  slow/HSBULT16_BUF_14        1.1923      0.0055      2.1943
  slow/HSBULT16_BUF_D_16      1.1405      0.0080      2.1914
  slow/HSBULT16_BUF_12        1.0368      0.0055      2.1943
  slow/HSBULT16_BUF_S_12      1.0368      0.0054      2.1943
 *slow/HSBULT16_BUF_U_1       0.2074      0.0053      2.1942
```

In the following example, the *estimate_eco* command analyzes the timing of the path that starts from register clock pin reg8/CP and ends at pin U3/A, considering only the rising_to_rise arc sense from the register clock pin to the register output by means of the -through_arc option.

```
pt_shell> set my_arc [get_timing_arcs -from reg8/CP -to reg8/Q \\
  -filter "sense == rising_to_rise"]
  ...
pt_shell> estimate_eco reg8 -from reg8/CP -to U3/A -through_arc $my_arc
```

The following example shows how to evaluate the data_pin_slack for clock-gating enable pin.

```
pt_shell> set eco_estimation_output_columns "arrival slack
 data_pin_slack";
pt_shell> estimate_eco cg1/EN -data_pin -max -rise ;

  delay type : max_rise
  lib cell                          arrival     slack     data_pin_slack
  ------------------------------------------------------------------------
  slow/HSBULT16_CKGTPLT_V7_16       0.0339      9.9598      2.1871
  slow/HSBULT16_CKGTPLT_V7_12       0.0308      9.9629      2.1885
  slow/HSBULT16_CKGTPLT_V7_8        0.0268      9.9670      2.1928
  slow/HSBULT16_CKGTPLT_V5_12       0.0398      9.9531      2.1969
 *slow/HSBULT16_CKGTPLT_V7_6        0.0257      9.9681      2.1942
  slow/HSBULT16_CKGTPLT_V5_8        0.0346      9.9587      2.1967
  slow/HSBULT16_CKGTPLT_V7_4        0.0246      9.9692      2.1946
  slow/HSBULT16_CKGTPLT_V5_6        0.0367      9.9564      2.1965
  slow/HSBULT16_CKGTPLT_V5_4        0.0333      9.9601      2.1967
  slow/HSBULT16_CKGTPLT_V5_2        0.0303      9.9633      2.1975
```

```
  slow/HSBULT16_CKGTPLT_V7_1      0.0242      9.9690      2.1949
  slow/HSBULT16_CKGTPLT_V5_1      0.0326      9.9606      2.1975
  slow/HSBULT16_CKGTPLT_V7_2      0.0229      9.9706      2.1952
```

The following example shows how the slack value can be parsed from any *estimate_eco* verbose report for a given min/max rise/fall condition:

```
# specify lib_cell and design instance
set lib_cell INVX20
set instance wishbone/bd_ram/U9/U6200

# obtain all min/max rise/fall estimation info for a single ECO
redirect -variable rpt {estimate_eco -type size_cell -min -max \\
  -rise -fall -verbose -lib_cells $cell $instance}

# specify min/max rise/fall conditions of interest
set mm max
set rf rise

# obtain min/max rise/fall subreport
regexp "(${mm}_${rf}.+?)\\n\\n" $rpt dummy subrpt

# obtain slack from subreport
regexp {slack +[^ ]+ +([^ ]+)} $subrpt dummy slack
echo $slack
```

This report parsing example can be used with both *size_cell* and *insert_buffer* estimation.

See Also

- get_alternative_lib_cells
- report_constraint
- report_delay_calculation
- report_timing
- eco_estimation_output_columns

---

estimate_memory_max_power

Find the worst-case internal energy when-condition in the liberty model for memory cells and set those for each of instances.

Syntax

```
int estimate_memory_max_power
[-pin pin_name_list]
[-dual_port]
[-state_derate_pairs]
```

instance_list
state_derate_pairs_list

Data Types

```
pin_name_list           list
instance_list           list
state_derate_pairs_list list
```

Arguments

`-pin` *pin_name_list*

To specify the input pin (or pins for -dual_port) which needs to be used for derating and power calculation.

`-dual_port`

To specify if memory cell is of dual port. Both the input pins must be passed in -pin option for this case.

instance_list

To specify the objects for which the maximum power estimation is to be done.

state_derate_pairs_list

To specify the list of when condition/regular expression and corresponding derate values.

-state_derate_pairs

To specify the list of mode for read or write states and corresponding derate values. The mode for read or write states comes from memory config file.

Description

Find the worst-case internal energy when-condition in the liberty model and set those for each of instances. Ignore all the propagated TR and SP values to the nets which are part of the when-condition, if there is a logic conflict with the setting, it is ignored. For all other pins which are not part of the selected when-condition use the propagated TR and SP.

This command should only be run in the average mode. update_power should be used to calculate power and report_power shows the maximum power values used for memory cells.

The *pin* option can be used to specify the input pin(or pins for -dual_port) which needs to be used for derating and power calculation.

Examples

The following examples shows how to provide when condition/regular expression and corresponding derate values on a single cell in the design for power estimation.

Feedback

```
pt_shell> estimate_memory_max_power I_SINGLE_PORT {"CS & !WE & !PD &
 SLEEPB" 0.5
"CS & WE & !PD & SLEEPB" 0.5} -pin "CK"
DB Schema Error : Class atom attribute when was declared in DB schema to
 be
type db_int created attribute of type db_string.
arc_id is 1, from_pin is (null), to_pin is CK, when_state is CS & !WE
 & !PD & SLEEPB
arc_id is 0, from_pin is (null), to_pin is CK, when_state is CS & WE
 & !PD & SLEEPB
Sum of matching states derate factors is 1.000000 <= 1.0
Total matching states power (after derating): 52.0000
Worst non matching states power: 3.1000
Worst non matching states power (after derating of 0.000000): 0.000000
Total (matching + non matching states) power is 52.000000
VL flow: For cell: I_SINGLE_PORT setting memory cell max power to 52.0000
1

pt_shell>estimate_memory_max_power I_SINGLE_PORT {"CS & *" 0.7, "WE & *"
 0.5}
-pin "CK"
Overlapping operations
Derate applied to power of worst overlapping state CS & WE & !PD & SLEEPB
 is 0.500000
Derated power for this state is 29.0000

Derate applied to power of first input regex state CS & !WE & !PD &
 SLEEPB is 0.200000
Derated power for this state is 9.2000

Derate applied to power of worst non-matching state !CS & !PD & SLEEPB is
 0.300000
Derated power for this state is 0.9300

Total power is 39.130001
VL flow: For cell: I_SINGLE_PORT setting memory cell max power to 39.1300
1

pt_shell>estimate_memory_max_power I_DUAL_PORT -pin {ACK BCK}
{"ACS & *" 0.5, "BCS & *" 0.7} -dual_port

Power estimation with clock pin : ACK
Sum of matching states derate factors is 0.500000
Total matching states power (after derating): 32.0000
Worst non matching states power: 3.6500
Worst non matching states power (after derating of 0.500000): 1.8250
Total (matching + non matching states) power is 33.825001

Power estimation with clock pin : BCK
Sum of matching states derate factors is 0.700000
Total matching states power (after derating): 49.0000
Worst non matching states power: 3.9800
```

```
Worst non matching states power (after derating of 0.300000): 1.1940
Total (matching + non matching states) power is 50.194000

VL flow: For cell: I_DUAL_PORT setting memory cell max power to
 84.0190(33.8250 + 50.1940)
1
```

The following examples shows how to provide state_derate_pairs specify the list of READ/WRITE modes and corresponding derate values. The modes comes form configuration file.

```
pt_shell>estimate_memory_max_power estimate_memory_max_power top/inst
 -state_derate_pairs {PWL_READ 0.4 PWL_WRITE 0.6}
```

See Also

- update_power
- report_power
- set_emmp_configuration

---

exit

Terminates the application.

Syntax

```
integer exit
[exit_code]
```

Data Types

```
exit_code      integer
```

Arguments

exit_code

Specifies the return code to the operating system. The default value is 0.

Description

This command exits from the application. You have the option to specify a code to return to the operating system.

Examples

The following example exits the current session and returns the code 5 to the operating system. At a UNIX operating system prompt, verify (*echo*) the return code as shown.

Feedback

```
prompt> exit 5

% echo $status
5
```

See Also

- quit

---

extract_model

Generates an abstract timing model from the gate-level netlist of the current design.

Syntax

status *extract_model*

```
[-context_borrow]
[-latch_level levels]
-output file_name
[-format format_list]
[-script_format ptsh | dcsh]
[-library_cell]
[-ignore_boundary_parasitics]
[-test_design]
[-validate valid_list]
[-noise]
[-power]
[-sms_scenarios sms_scenarios_list]
```

Data Types

levels	int
file_name	string
format_list	list
valid_list	list
arc_list	list
sms_scenarios_list	collection

Arguments

`-context_borrow`

By default, model extraction considers the borrowing behavior of transparent latches on the design boundary based on the current timing context. Using the *-context_borrow* option is not required and has no effect.

`-latch_level` *levels*

Specifies the maximum number of levels of latches considered for extraction at the interface of the design, either 0, 1, or 2. During model extraction, the tool traces through a transparent latch and stops at a latch/register, up to the number

of levels specified by this option. If you specify a number larger than 2, that number is accepted as valid syntax, but the extraction internally considers no more than 2 levels. If the *extract_model_include_transparent_latch_constraints* variable has been set to *false*, the extraction internally considers no more than 1 level.

`-output` *`file_name`*

Specifies the root name to generate output file names. The generated model name is *file_name*_lib.db (compiled library format) or *file_name*.lib (Liberty format). The generated constraint script file name is *file_name*_constr.pt.

`-format` *`format_list`*

Specifies the output formats for the generated model. Multiple items must be enclosed in braces. The allowed values are:

- *db* (the default) - Generates the output file in compiled library format with the _lib.db extension.
- *lib* - Generates the output file in Liberty (Library Compiler) format with the .lib extension.

`-script_format ptsh | dcsh`

Specifies the output format for the constraint script. Allowed values are *ptsh* for pt_shell (the default) or *dcsh* for dc_shell.

`-library_cell`

Generates a library cell. The option specifies the default behavior, so it is not needed and has no effect.

`-ignore_boundary_parasitics`

Excludes the effects of detailed parasitics on boundary (port-connected) nets from the extracted timing model.

`-test_design`

Generates a test design containing an instance of the extracted timing model as well at the extracted model itself. The extracted timing model name is *file_name*_lib.db and the test design name is *file_name*_test.db, where *file_name* is specified by the *-output* option.

`-validate` *`valid_list`*

Specifies the type of data automatically validated after the model is created, *timing*. The validation process is performed automatically only when you use this option. Before you use this option, set the *hier_modeling_version* variable to *2.0*. The validation process uses the *write_interface_timing* and *compare_interface_timing* commands.

`-noise`

Adds noise features to the extracted model. If no noise information is available, only the estimated steady-state resistance is written to the model. This option performs an implicit *update_noise* if needed.

`-power`

Adds power features to the extracted model. This option performs an implicit *update_noise* if needed. The *power_enable_analysis* variable must be set to *true*, and you must have a PrimePower license.

`-sms_scenarios` *sms_scenarios_list*

Restricts extraction to the specified SMVA/SMS scenarios. This collection is created by the *get_sms_scenarios* command. The default is to create a model for all SMS scenarios relevant to the design.

In an SMVA/SMS analysis, the *extract_model* command creates a separate model for each SMS scenario, augmenting the file name with a description of the voltage configuration as follows:

```
<name>_<supply_group1>_<ref_name1>[..._<supply_groupN>_<ref_nam
eN>]
```

where *supply_group1,.,supply_groupN* are the supply groups included in the voltage configuration, and *ref_name1,..ref_nameN* are their corresponding voltage reference names specified with the *set_voltage_levels* command.

Description

The *extract_model* command generates a static timing model for the current design from its gate-level netlist. The generated model exhibits the same timing characteristics as the original design and therefore is used instead of the gate-level netlist for timing analysis at a higher level of the design hierarchy. To also generate noise or power information in the model, use the *-noise* or *-power* option.

By default, the *extract_model* command generates a context-independent model, in which the timing changes as the actual context varies at the top level. The timing model tracks the original block interface timing. However, the extracted model can become context-dependent. Especially when borrowing latches exist at the block interface, the generated model is valid only for the clock waveforms, latencies, input delays, and output delays specified for the extraction. When the context changes sufficiently to cause different borrowing on the interface, the timing model does not represent the new borrowing.

To generate a timing model for a design, you need the design netlist, the logic library, and the timing environment (clocks, constraints, and operating conditions). There should be

no timing violations reported by the *check_timing* and *report_constraint* commands. The generated model is captured as a library cell in .db or .lib format, or both.

The following variables are often used to control various aspects of the model extraction process:

```
extract_model_capacitance_limit
extract_model_clock_transition_limit
extract_model_data_transition_limit
extract_model_enable_report_delay_calculation
extract_model_gating_as_nochange
extract_model_num_capacitance_points
extract_model_num_clock_transition_points
extract_model_num_data_transition_points
extract_model_status_level
extract_model_with_ccs_timing
extract_model_with_clock_latency_arcs
timing_clock_gating_propagate_enable
timing_disable_clock_gating_checks
timing_enable_preset_clear_arcs
```

For a complete list:

```
pt_shell> printvar extract_model*
```

Examples

The following script reads in a design, links it, prepares it for model extraction, and extracts a model for the design, producing a .db library file.

```
set_app_var search_path ". /abc/libraries/mylib"
set_app_var link_path "* 1.3V_0C.db"
read_verilog mydesign.v
link_design mydesign
read_parasitics mydesign.spef
source myconstraints.sdc
report_constraint
check_timing
report_timing
set_app_var extract_model_capacitance_limit 5.0
extract_model -output mymodel
```

The *extract_model* command responds with the following messages:

```
pt_shell> extract_model -output mymodel
Model extraction : Clock period/min_pulse_width arcs extracted at context
 slew.
Wrote model to './ETM/mymodel/mod_lib.db'
Wrote the propagated generated clocks in ./ETM/mymodel/mod_inst.pt.gz.
Wrote generated clock uncertainty in ./ETM/mymodel/mod_inst.pt.gz.
1
```

Feedback

The following command generates an extracted timing model in .lib format, in addition to .db format. You can read the .lib file to examine the extracted timing arcs.

```
pt_shell> extract_model -output mymodel -format {db lib}
Model extraction : Clock period/min_pulse_width arcs extracted at context
 slew.
Wrote the LIB file ./ETM/mymodel/mod.lib
Wrote model to './ETM/mymodel/mod_lib.db'
 ...
```

The following command generates a test design containing an instance of the generated timing model.

```
pt_shell> extract_model -output mymodel -test_design
Model extraction : Clock period/min_pulse_width arcs extracted at context
 slew.
Wrote model to './ETM/mymodel/mod_lib.db'
Wrote test design to './ETM/mymodel/mod_test.db'
 ...
```

The following example generates a timing model and also performs timing validation of the generated model.

```
pt_shell> set_app_var hier_modeling_version 2.0
2.0
pt_shell> extract_model -output mymodel -validate timing
Model extraction : Clock period/min_pulse_width arcs extracted at context
 slew.
Wrote model to './ETM/mymodel/mod_lib.db'
Wrote the propagated generated clocks in ./ETM/mymodel/mod_inst.pt.gz.
Wrote generated clock uncertainty in ./ETM/mymodel/mod_inst.pt.gz.
 ( Worker2 )
Information: all model validation immediate outputs are stored in
 directory
 '/remote/myworkingdir/model_validation'. (MV-013)
Information: Starting to launch validation session ...
Information: Done launch of validation session.
Information: Starting write_interface_timing for model ...
 ...
Information: Done write_interface_timing for model.
Information: Starting compare_interface_timing ...

        Totals  Slack     Transition Time   Capacitance  Rules
-------------------------------------------------------------
Passed  716     0         164               552          -
Failed  736     624       112               0            0
Total   1452    624       276               552          0

Information: Done compare_interface_timing.
Information: Starting pba ...
Information: Done pba.
Information: Starting merge report ...
```

```
Information: Done merge report.
Information: Starting to compare interface timing after pba ...

        Totals  Slack     Transition Time   Capacitance  Rules
----------------------------------------------------------------
Passed  716     0         164               552          -
Failed  736     624       112               0            0
Total   1452    624       276               552          0

Information: Done compare interface timing after pba.
Shutting down worker processes ...
 Shutdown Process 2
1
```

See Also

- compare_interface_timing
- write_interface_timing
- get_sms_scenarios
- set_voltage_levels
- extract_model_capacitance_limit
- extract_model_clock_latency_arcs_include_all_registers
- extract_model_clock_transition_limit
- extract_model_create_variation_tables
- extract_model_data_transition_limit
- extract_model_db_naming_compatibility
- extract_model_enable_report_delay_calculation
- extract_model_gating_as_nochange
- extract_model_include_clock_tree_pulse_width
- extract_model_include_ideal_clock_network_latency
- extract_model_include_transparent_latch_constraints
- extract_model_include_upf_data
- extract_model_keep_inferred_nochange_arcs
- extract_model_lib_format_with_check_pins
- extract_model_noise_iv_index_lower_factor

Feedback

- extract_model_noise_iv_index_upper_factor
- extract_model_noise_width_points
- extract_model_num_capacitance_points
- extract_model_num_clock_transition_points
- extract_model_num_data_transition_points
- extract_model_num_noise_iv_points
- extract_model_num_noise_width_points
- extract_model_short_syntax_compatibility
- extract_model_single_pin_cap
- extract_model_single_pin_cap_max
- extract_model_split_partial_clock_gating_arcs
- extract_model_status_level
- extract_model_suppress_three_state
- extract_model_upf_supply_precedence
- extract_model_with_3d_arcs
- extract_model_with_ccs_timing
- extract_model_with_clock_latency_arcs
- extract_model_write_case_values_to_constraint_file
- extract_model_write_verilog_format_wrapper
- timing_clock_gating_propagate_enable
- timing_disable_clock_gating_checks
- timing_enable_preset_clear_arcs

---

extract_scaling_data_config

Extract scripts to generate scaling data file setup.

Syntax

string *extract_scaling_data_config*

Feedback

```
[-creation]
[-consumption]
```

Arguments

`-creation`

Script to create scaling data files.

`-consumption`

Script to replace define_scaling_library_group commands in original script.

Description

The *extract_scaling_data_config* command can be used to extract commands to use the scaling_data file flow from an exiting design setup. Run your existing script up until update_timing then 1.) Call *extract_scaling_data_config -creation*. This will generate all commands needed to generate the scaling data files needed for the new flow. 2.) Call *extract_scaling_data_config -consumption*. This will generate a sccript with define_library_scaling_group_commands to use the scaling data files created. Replace the define_library_scaling_group_commands commands in the original script with these newly created ones and the original script will now run the scaling data file flow.

Examples

The following snippet would typically be used after update timing to extract commands scripts for creating scaling data files setup.

```
pt_shell> extract_scaling_data_config -creation
```

f

filter

The *filter* command, a synonym for the *filter_collection* command, is a Design Compiler emulation command provided for compatibility between PrimeTime and Design Compiler. You use the *filter_collection* command to filter an existing collection, resulting in a new collection. The base collection remains unchanged.

See Also

- filter_collection

Feedback

filter_collection

Filters an existing collection, resulting in a new collection. The base collection remains unchanged.

Syntax

collection *filter_collection*

```
[-regexp]
[-nocase]
collection1
expression
```

Data Types

collection1	collection
expression	string

Arguments

`-regexp`

Specifies that the =~ and !~ filter operators will use real regular expressions. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the * and ? wildcards.

`-nocase`

Makes the pattern match case-insensitive.

collection1

Specifies the base collection to be filtered. This collection is copied to the result collection. Objects are removed from the result collection if they are evaluated as *false* by the conditional *expression* value. Substitute the collection you want for *collection1*.

expression

Specifies an expression with which to filter *collection1*. Substitute the string you want for *expression*.

Description

Filters an existing collection, resulting in a new collection. The base collection remains unchanged. In many cases, application commands that create collections support a *-filter* option that filters as part of the collection process, rather than after the collection has been made.

Feedback

This type of filtering is almost always more efficient than using the *filter_collection* command after a collection has been formed. The *filter_collection* command is most useful if you plan to filter the same large collection many times using different criteria.

The *filter_collection* command results in either a new collection or an empty string. A resulting new collection contains the subset of the objects in the input *collection1*. A resulting empty string (the empty collection) indicates that the *expression* filtered out all elements of the input *collection1*.

The basic form of the conditional expression is a series of relations joined together with AND and OR operators. Parentheses () are also supported. The basic relation compares an attribute name with a value through a relational operator.

For example,

```
is_hierarchical == true and area <= 6
```

If an attribute is a collection attribute, then you can query an attribute value from that object using "." to name the attribute on the other object. The "." operator can be chained. If the collection attribute refers to more than one object, then the expression is true if it is true for one of the objects referenced. While "." is preferred, using "->" is also supported.

For example:

```
owner.is_hierarchical == true and owner.area <= 6
shape.net.name==reset
```

The value side of a relation can be a simple string, quoted string, or an attribute name prefixed with "@". For example:

```
input_delay<=@output_delay
input_delay<=0.24
name=="@literal_name"
```

The relational operators are

```
==   Equal
!=   Not equal
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
=~   Matches pattern
!~   Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only either *true* or *false*.

Existence relations determine if an attribute is defined or not defined for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
!defined
```

These operators apply to any attribute as long as it is valid for the object class.

Collection attributes support a special sizeof(attrName) operator that returns the number of objects in the attribute. If the attribute is not set then 0 is returned.

The *filter_collection* command has a *-regexp* option that uses regular expressions when matching for string attributes. Regular expression matching is done in the same way as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed. You can make the regular expression search case-insensitive by using the *-nocase* option.

Mathematical expressions are supported. All function names must be prefixed with the '#' in order to disambiguate the function names from subscripted attribue names (i.e. "#sin(attr)"). In order to use a math expression on the right hand side of a comparison, the expression must be contained in parenthesis. This is required for backward compatibility with the legacy expression syntax as strings on the right do not require quoting.

Subscripted attributes can match against glob style subscript patterns (i.e. "input_delay(*)<1.0"). In that case the term evaluates to true if any subscript satisfies the condition.

Filters may also refer to chained attributes that may imply multiple objects in a collection (i.e. "pins.capacitance==0.2"). In that case the term returns true if any of the objects referred to in the have an attribute value satisfying the condition.

Feedback

Examples

The following example from PrimeTime creates a collection of only hierarchical cells.

```
pt_shell> set a [filter_collection [get_cells *] \\
"is_hierarchical == true"]
{"Adder1" "Adder2"}
```

The following example from PrimeTime uses existence operators to create a collection of all nonmoded cell timing_arc objects in the current design.

```
pt_shell> set b [filter_collection \\
            [get_timing_arcs -of_objects [get_cells *]] \\
            "undefined(mode)"]
```

```
The following example finds all cells from {U0(1) U1(1) U0(0) U1(0)}
with the index (0). Notice the round brackets used for the index.

pt_shell> filter_collection -regexp [get_cells] {full_name =~
 "U.*\\[0\]\)"}
{U0(0) U1(0)}
```

The following example finds all ports from in[18] to in[23]. The placement of double quotes and backslashes is necessary to avoid syntax errors.

```
pt_shell> filter_collection [get_ports] -regexp {name=~
 "in\\[(1[8-9]|2[0-3])\\]"}
{in[23] in[22] in[21] in[20] in[19] in[18]}
```

The following example finds the U2 and U4 cells, using the -nocase option:

```
pt_shell> filter_collection [get_cells] -regexp -nocase {name =~ u[24]}
{U2 U4}
```

See Also

- collections

---

find

The *find* command, used to create a collection of design objects, is a DC Emulation command provided for compatibility with Design Compiler.

Syntax

string *find*

```
type
[-hierarchy]
[-flat]
[object_list]
```

Feedback

Data Types

type	string
object_list	list

Arguments

type

Specifies the object class to find. The type value can be *design*, *port*, *net*, *cell*, *pin*, *clock*, *library*, *lib_cell*, or *lib_pin*.

-hierarchy

Find objects throughout the hierarchy.

-flat

Not supported by PrimeTime.

object_list

Specifies the patterns to match.

Description

The *find* command creates a collection of objects. The command exists in PrimeTime for compatibility with Design Compiler. Complete information about the *find* command can be found in the Design Compiler documentation. The supported method for creating collections of objects is to use *get** commands (for example, the *get_cells* command or the *get_pins* command).

Examples

The following example finds the N1 net in the current design:

```
pt_shell> find net N1
{N1}
```

The following example uses the * (asterisk) wildcard character to find all cells in the current design that begin with U:

```
pt_shell> find cell U*
{U0, U1, U2, U3, U4, U5, U6, U7}
```

For more examples, see the *find* command in Design Compiler.

See Also

- get_cells
- get_clocks
- get_designs

Feedback

- get_lib_cells
- get_lib_pins
- get_libs
- get_nets
- get_pins
- get_ports

find_objects

Finds logical hierarchy objects within a scope. This is a UPF query command.

Syntax

```
[-collection find_objects]
[-pattern pattern_string]
[-object_type inst | port | net | model]
[-direction in | out | inout]
[-transitive true | false]
[-non_leaf]
[-leaf_only]
[-regexp]
[-exact]
scope
```

Data Types

```
pattern_string      string
scope               string
```

Arguments

`-pattern pattern_string`

Specifies a search pattern. By default, the pattern is treated as an Tcl glob expression.

`-object_type inst | port | net | model`

Searches for only the specified object type:

- *inst* - Cell instance • *port* - Design port • *net* - Net • *model* - Model or library cell

`-direction in | out | inout`

Specifies the direction of the searched ports if the *-object_type port* option is used.

`-transitive true | false`

Controls whether the descendants of the elements are included in the search:

- *false* (default) - Does not search descendants. • *true* - Searches descendants.

`-non_leaf`

Searches only nonleaf design elements (elements that have child elements).

`-leaf_only`

Searches only leaf-level design elements (elements without child elements). By default, both leaf and nonleaf design elements are searched.

`-regexp`

Views the patterns argument as real regular expressions rather than simple wildcard patterns.You can widen the search by adding ".*" to the beginning or end of the expressions as needed. The -regexp and -exact options are mutually exclusive.

`-exact`

Searches for an exact match of the *pattern_string* argument; disallows wildcard expansion on the *pattern_string*.

scope

Specifies the scope in which to search for logical hierarchy objects.

Description

The *find_objects* command finds logical hierarchy objects within a scope.

Multicorner-Multimode Support

This command has no dependency on scenario specific information.

Examples

The following examples show the usage of the command with different options.

```
prompt> find_objects a -pattern * -object_type inst \\
         -transitive TRUE -non_leaf
a/b1 a/b2

prompt> find_objects a/b2 -pattern b_? -object_type net
a/b2/b_i a/b2/b_o

prompt> find_objects a/b1 -pattern * -object_type port \\
        -transitive TRUE -direction out
a/b1/c1/Z a/b1/c2/Z a/b1/b_o
```

Feedback

```
prompt> find_objects . -object_type model -pattern b* -non_leaf \\
        -transitive TRUE
a/b1 a/b2 b1

prompt> find_objects . -object_type model -pattern b* -non_leaf \\
        -transitive TRUE -exact
Warning: Can't find module or lib cell 'b*' in design 'top'. (UID-95)

prompt> find_objects . -object_type model -pattern bot -non_leaf \\
        -transitive TRUE -exact
a/b1 a/b2 b1

For black box bus port
prompt> find_objects . -pattern A* -object_type port
A[0] A[1] A[2]
```

See Also

- get_cells
- get_nets
- get_ports
- set_scope

---

fix_eco_drc

Fixes or improves design rule constraint (DRC) violations (noise violations, large delta delays, or cell electromigration violations) by sizing cells or inserting buffers or inverter pairs.

Syntax

```
status fix_eco_drc
   -type max_transition | max_capacitance | max_fanout | noise |
 delta_delay | cell_em
   [-methods method_list]
   [-buffer_list list]
   [-verbose]
   [-current_library]
   [-physical_mode none | open_site | occupied_site | freeze_silicon]
   [-hold_margin margin]
   [-setup_margin margin]
   [-cell_type combinational | clock_network]
   [-slack_lesser_than slack_limit]
   [-unfixable_reasons_format text | csv]
   [-unfixable_reasons_prefix string]
   [-delta_delay_threshold threshold]
   [-timeout seconds]
   [object_list]
```

Feedback

Data Types

list	list
margin	float
method_list	list
object_list	list
seconds	integer
slack_limit	float
string	string
threshold	float

Arguments

`-type max_transition | max_capacitance | max_fanout | noise | delta_delay | cell_em`

Specifies the type of violation to fix: maximum transition time, maximum total net capacitance, or maximum fanout DRC violations, as reported by the *report_constraint* command; or noise violations, as reported by the *report_noise* command; or large delta delays, as reported by the *report_si_bottleneck -cost_type delta_delay* command; or cell electromigration violations, as reported by the *report_cell_em_violation* command. Use multiple *fix_eco_drc* commands to fix multiple types of violations.

Note: The *cell_em* value requires PrimePower and PrimeTime-ADV-PLUS licenses. The *delta_delay* value requires a PrimeTime-ADV-PLUS license.

`-methods` *method_list*

Specifies one or more fixing methods from the following list:

- *size_cell* - Replaces cells in the timing path with logically identical cells that have a different drive strength.
- *insert_buffer* - Inserts buffers in the timing path, using the library cells specified by the *-buffer_list* option. By default, the *insert_buffer* method inserts buffers at both driver and load pins and on-route.
- *insert_buffer_at_load_pins* - Inserts buffers only at load pins, not at driver pins and the middle of timing path.
- *insert_buffer_at_driver_pins* - Inserts buffers only at driver pins, not at load pins and the middle of the timing path.
- *insert_inverter_pair* - Inserts inverter cell pairs instead of buffer cells; cannot be used together with the *insert_buffer* method.

The default is *-methods {size_cell insert_buffer}.*

`-buffer_list` *list*

Specifies the list of library cell buffers that can be used for fixing with the *insert_buffer* method, or inverters that can be used with the *insert_inverter_pair* method. To use the *insert_buffer* or *insert_inverter_pair* method, you must also use the *-buffer_list* option to specify the library cells to be used; there is no default list.

Specify the list of buffers using simple library cell base names, without the library name. The command gets the buffer cells from the libraries listed in the *link_path* variable. It searches the libraries in order and uses the first library cell found with a matching name.

The command can use a listed library cell for buffer insertion even if its *dont_use* attribute is set to *true*. The *dont_use* attribute applies only to cell sizing.

If both the *size_cell* and *insert_buffer* methods are being used, the command can insert a buffer from the list and then size it to a different buffer, even if the sized library cell is not explicitly included in the *-buffer_list* list.

`-verbose`

Shows additional information during the fixing process. In distributed multi-scenario analysis, violation information from each scenario is displayed.

The verbose mode also generates an "Unfixable Violations" report upon completion of fixing. The report shows a list of pins where fixing was considered and the reasons that the fix was not performed at the pin. This option is recommended for interactive ECO because it can help you decide what to do next. For details, see "Unfixable Violation Report" in the DESCRIPTION section of this man page.

`-current_library`

Specifies the usage of library cells from the same library when performing cell sizing. With this option, the replacement cell must come from the same library as the original cell. This option can improve runtime because fewer cells are considered as potential replacements. However, it might degrade quality of results because the variety of available alternative library cells could be reduced.

`-physical_mode none | open_site | occupied_site | freeze_silicon`

Specifies the usage of physical data to size cells or place buffers:

- *none* (the default) - DRC fixing does not use physical data.
- *open_site* - DRC fixing sizes a cell only if there is enough room available around the cell and inserts a buffer only if there is an empty site with enough room to accept the new cell without moving nearby cells. This mode retains the placement of existing cells.

- *occupied_site* - DRC fixing can size a cell and insert a buffer that overlaps existing neighbor cells, as long as the cell density (area utilization) is low enough that the nearby cells can be moved to make the required space. After the change is made, you need to use a physical implementation tool such as IC Compiler II to move the existing cells and create room for the sized or inserted cell.
- *freeze_silicon* - DRC fixing uses spare cells defined in the physical data files and identified by the *-programmable_spare_cell_names* option of the *set_eco_options* command; it does not insert new cells or size existing cells. This mode works with the IC Compiler II tool.

`-hold_margin` *margin*

Specifies a margin applied to hold timing slack during DRC fixing, in library time units. DRC fixing tries (but does not guarantee) to preserve the specified amount of hold timing slack. For example, a margin of zero seeks to prevent hold timing violations while allowing all positive hold slack to be given up for DRC fixing. By default, without this option, DRC fixing proceeds no matter how much hold timing is degraded, which is equivalent to a hold margin of minus infinity.

`-setup_margin` *margin*

Specifies a margin applied to setup timing slack during DRC fixing. DRC fixing tries to preserve the specified amount of setup timing slack. By default, DRC fixing proceeds no matter how much setup timing is degraded.

In the absence of explicit setup and hold values provided by the user, the tool tries to preserve DRC and timing by default, however, the results may not match explicit settings of *-setup_margin 0.0* and *-hold_margin 0.0*.

`-cell_type combinational | clock_network`

Specifies the types of cells modified during fixing, either *combinational* (the default) or *clock_network*. For the *combinational* setting, fixing is performed by sizing or inserting cells in data paths. For the *clock_network* setting, fixing is performed by sizing or inserting cells in the clock network. Changing clock network cells can affect clock tree timing, so use this option only when absolutely necessary.

`-slack_lesser_than` *slack_limit*

Specifies fixing of only the paths with a slack worse than the specified slack limit. The default is zero, which attempts to fix all DRC violations. The limit can be set to a positive number only if the *-type* option is set to *delta_delay*. In that case, the command works like the *report_si_bottleneck* command to select delta delay victim nets on noncritical paths.

`-unfixable_reasons_format text | csv`

Specifies the file format, either *text* or *csv* (comma-separated values), for the unfixable reasons report generated by the *-unfixable_reasons_prefix* option. The CSV report format is compatible with the PrimeTime GUI and spreadsheet programs. By default, both types of files are generated when the *-unfixable_reasons_prefix* option is used.

`-unfixable_reasons_prefix` *`string`*

Specifies a prefix used for naming the file generated by the *-unfixable_reasons_format* option. For example, specifying the prefix "abc" and using the CSV format writes out a file named abc_eco_drc.csv. If you omit this option, no files are generated.

`-delta_delay_threshold` *`threshold`*

Specifies the delta delay threshold for delta delay fixing. Fixing applies only the victim nets that belong to one or more paths with negative timing slack, and whose delta delay cost exceeds the specified threshold. The threshold must be greater than zero; it applies to delta delays in both directions (positive for max delays and negative for min delays). Using this option is mandatory when the *-type* option is set to *delta_delay*.

`-timeout` *`seconds`*

Specifies a timeout limit, in seconds, for the maximum total runtime of the command. At the end of each fixing iteration, the tool checks the total elapsed wall clock time. If the time limit is reached, the command stops running and ends with the current iteration. If you do not use this option, the command does not directly consider the total elapsed time. Instead, it considers quality of results, fix rate, and expected benefits of running more iterations.

`object_list`

Specifies a list of pin or port objects in the current design for DRC fixing, noise fixing, or delta delay reduction. If you do not specify an object list, the tool tries to fix all DRC or noise violations in the current design, or reduce delta delay on victim nets according to *-delta_delay_threshold* option.

Description

The *fix_eco_drc* command fixes or improves design rule constraint (DRC) violations, noise violations, large delta delays, or cell electromigation violations by sizing cells and inserting buffers or inverter pairs.

You can report existing violations by using the *report_constraint*, *report_noise*, *report_si_bottleneck -cost_type delta_delay*, or *report_cell_em_violation* command. The *fix_eco_drc* command attempts to fix the violations while minimizing the impact on area. By default, the command performs fixing without considering the effects on timing.

After the violations are fixed, you can write out the design changes as a script by using the *write_changes* command. You can use the script to implement the same changes in another PrimeTime session or another tool (Design Compiler, IC Compiler, or IC Compiler II). A list of design changes created in this flow is called an engineering change order (ECO).

The *fix_eco_drc* command has options to specify the following fixing parameters:

- The type of violation to fix (maximum transition time, maximum total net capacitance, maximum fanout, noise, delta delays, or cell electromigration that exceed a specified threshold and contribute to negative slack)
- The scope of the design to be fixed (a list of pins or ports, or the whole design)
- The fixing methods (cell sizing, buffer insertion, or inverter pair insertion)
- The list of library cells that can be used for buffer or inverter pair insertion
- The types of cells to be modified (cells in data paths or cells in clock networks)
- Whether to consider timing constraints during fixing, and if so, the required setup and hold timing margins
- The physical placement and sizing mode (none, open site, occupied site, or freeze silicon)

You must specify the type of violation to fix by setting the *-type* option to *max_transition*, *max_capacitance*, *max_fanout*, *noise*, *delta_delay*, or *cell_em*. By default, the command fixes violations of the specified type throughout the design. You can restrict fixing to a specified list of pins and ports.

Note: The *cell_em* fixing type requires that the *power_enable_analysis* and *power_enable_em_analysis* variables be set to *true*.

The command performs fixing using multiple iterations. It repeats these iterations until all violations are fixed or it determines that further fixing is not worth the runtime cost, based on the current quality of results and fixing option settings. You can generate a report on unfixable violations by using the *-verbose* option. To write out the unfixable violation report to a file, use the *-unfixable_reasons_prefix* option.

Note: The *cell_em* fixing type supports only a single iteration. To fix additional violations, run the *update_power* command, then rerun the *fix_eco_drc -type cell_em* command again.

By default, the command fixes the violations without considering the effects on timing. To prevent new timing violations during fixing, use the options *-setup_margin 0.0* and *-hold_margin 0.0*; or specify positive margin values to maintain positive timing slack values, or specify negative margin values to allow timing violations with the specified amount of negative slack.

The *fix_eco_drc* command is compatible with single-core analysis, distributed multi-scenario analysis (DMSA), and multicore analysis. To ensure consistent results during DMSA fixing, apply any *dont_touch* and *dont_use* attributes identically to all scenarios.

The *fix_eco_drc* command is intended for the signoff flow using delay and slew calculation in the PrimeTime tool, based on extracted parasitic data. The *timing_save_pin_arrival_and_required* variable must be set to *true* to perform fixing. If delays and slews are annotated directly (for example, in SDF format), the quality of results cannot be guaranteed.

Fixing Methods

You can use the *-methods* option to specify the fixing methods, one or more of *size_cell*, *insert_buffer*, and *insert_inverter_pair*. The default is *-methods {size_cell insert_buffer}*.

If you specify only *size_cell* as the fixing method, the command resizes the driver cell of the stage with a violation. If a violation is at a cell input pin, a driver cell to this pin is resized. If a violation is at a cell output pin, the cell itself is resized. The selection of larger cells is restricted by the *eco_alternative_area_ratio_threshold* variable. By default, this variable is set to 2.0, which restricts the new cell to no more than twice the size of the original cell.

Only combinational logic cells are resized. Sequential cells are not resized because doing so could potentially unbalance clock trees and create unexpected new violations.

If you specify only *insert_buffer* or *insert_inverter_pair* as the fixing method, the command inserts buffers or inverter pairs to fix violations, breaking physically large nets into smaller nets.

If cell sizing and buffer insertion are both enabled (the default behavior), the command chooses the best fixing method to remove each violation and minimize the impact on area. In this case, cell sizing ignores the *eco_alternative_area_ratio_threshold* variable and uses cells of any size because more layout perturbation is allowed.

Restricting the Library Cells Used for Sizing

To prevent the tool from using specific library cells for sizing, apply the *dont_use* attribute to these library cells with the *set_dont_use* command. Note that this usage restriction applies only to cell sizing, not buffer insertion. The tool uses buffers specified by the *-buffer_list* option, irrespective of the *dont_use* attribute.

Usage of library cells can also be restricted by applying the *pt_dont_use* user-defined attribute on the restricted lib_cell objects, which is a legacy feature. For an example of how to apply this attribute, see the EXAMPLES section.

Feedback

Preserving Specific Cells and Nets During Fixing

To prevent the *fix_eco_drc* command from performing fixing on specific cells or nets, apply a *dont_touch* attribute to the cell or net. For example,

```
pt_shell> set_dont_touch [get_nets n123] true
```

The *fix_eco_drc* command does not size any cell with a *dont_touch* setting and does not insert a buffer or inverter pair on any net with a *dont_touch* setting.

If you apply a *dont_touch* setting to a hierarchical cell, that setting propagates downward to the child cells and nets of the hierarchical cell. You can override the *dont_touch* setting for specific lower-level cells and nets by setting the *dont_touch* attribute to *false* for those objects.

UPF Cells

By default, UPF cells are not considered for sizing. UPF cells are cells that have one or more of the following attributes set to *true*: *always_on*, *is_isolation*, *is_retention*, or *is_level_shifter*.

To allow UPF cells to be sized, set the *eco_allow_sizing_with_lib_cell_attributes* variable. For example,

```
pt_shell> set_app_var eco_allow_sizing_with_lib_cell_attributes \\
  {always_on is_isolation is_retention is_level_shifter}
```

A buffer inserted into a net must be placed in the same power domain as the driver and all loads of the net. Buffer insertion is not performed on the following nets:

- Nets connected to any pins of an always_on cell
- Nets connected to control and clock pins of a retention cell
- A net between a port and an isolation cell
- A net between a port and a level shifter cell

Note that the *fix_eco_drc* command allows always_on buffer insertion on nets connected with an always_on cell if *eco_allow_insert_buffer_always_on_cells* variable is set to *true* (false by default).

Timing Margins

By default, the *fix_eco_drc* command fixes DRC or noise violations even if the fixing creates new timing violations or worsens existing timing violations.

To prevent new timing violations, use the *-setup_margin 0.0* and *-hold_margin 0.0* options, or specify nonzero margins to trade off DRC or noise fixing against timing. A negative margin relaxes the timing constraints and allows DRC or noise fixing to proceed and

cause timing violations up to the specified negative slack. A positive margin reserves the specified amount of positive slack, allowing less freedom to fix DRC or noise violations.

Unfixable Violation Report

The *fix_eco_drc* command, upon completion of fixing, generates a report on unfixable violations when you use the *-verbose* option. Here is a report example:

```
pt_shell> fix_eco_drc -type max_transition -verbose \\
           -methods size_cell
 ...
Unfixable violations:
  A - There are available lib cells outside area limit
  C - The violation is in clock network
  I - Buffer insertion with given lib cells cannot fix the violation
  P - Driver cell of the violation is a port
  Q - Driver cell of the violation is a sequential cell
  S - Cell sizing with alternative lib cells cannot fix the violation
  T - Timing margin is too tight to fix the violation
  V - Net or cell is invalid or has dont_touch attribute
  U - UPF restricts fixing the violation

  Violation                                  Reasons
  -----------------------------------------------
  U1/I                                           P
  U6/I                                           S

Remaining Violations:
  Violation Type                               Count
  -----------------------------------------------
  Total remaining violations                       2
  Unfixable violations                             2
```

The "Violation" column shows the cell pins where unfixable violations are located.

The "Reasons" column shows the reason or reasons that a violation could not be fixed at the cell pin, using the single-letter codes from the key shown at the beginning of the report. Here are detailed descriptions of the reasons:

- *A - There are available library cells outside area limit*

  Library cells that might have been used to fix the violation exceeded the area limit specified by the *eco_alternative_area_ratio_threshold* variable. You might be able to fix the violation by increasing the area limit.

- *C - The violation is in clock network*

  The violation is in a clock network. By default, the command does not attempt to fix violations in clock networks because doing so can change the clock tree timing and possibly create new violations. If data path fixing is not successful (*-cell_type combinational*, the default), you can try clock network fixing (*-cell_type clock_network*).

- *D - Cell or net is located in high density area*

  There is not enough room physically available in the vicinity of the violation. In the physical implementation tool (such as IC Compiler II), check the area utilization in that location. Note that when the *eco_allow_filler_cells_as_open_sites* variable is set to *true* (the default), physically aware ECO commands treat filler cells as open sites.

- *E - Physical information is incomplete or unavailable*

  The physical information is incomplete or unavailable for the specific net or cell, or the net is marked as "+USE CLOCK" in the DEF file specified by the *set_eco_options* command. See if the DEF and LEF files are consistent with your current design.

- *H - Logical and physical hierarchy is inconsistent*

  The netlist and physical layout are not consistent, and a buffer cannot be inserted. Use the physical implementation tool to resolve the inconsistency.

- *I - Buffer insertion with given library cells cannot fix the violation*

  The violation cannot be fixed using the library cells listed by the *-buffer_list* option. Specifying a wider range of buffer cells might allow the command to fix the violation.

- *O - No open free site is available*

  There is no empty free site available on the route or in the vicinity of the violation. Inspect the LEF/DEF file for filler cells and see whether they are treated as empty free sites. When the *eco_allow_filler_cells_as_open_sites* variable is set to *true* (the default), physically aware ECO commands treat all filler cells as empty free sites. Also, you can use the physical implementation tool (such as IC Compiler II) to create empty free sites.

  If the *-physical_lib_constraint_file* option of the *set_eco_options* command has been used to specify intercell or advanced-node spacing rule constraints, some empty sites might not be wide enough to be used. In that case, you can use the physical implementation tool to examine the spacing rule constraints and the cells in the vicinity of such free sites.

- *Q - Driver cell of the violation is a sequential cell*

  The driver of the stage where the violation exists is a sequential cell, so resizing cannot be applied. The *fix_eco_drc* command does not resize sequential cells.

- *S - Cell sizing with alternative library cells cannot fix the violation*

  All alternative library cells have been considered for sizing to fix the violation, but the violation cannot be fixed by cell sizing. Use a buffer insertion method to fix the violation, if needed.

- *T - Timing margin is too tight to fix the violation*

The violation is on a critical or near-critical setup or hold timing path and there is not enough timing slack to allow fixing of the targeted violation. To allow fixing to proceed, you can relax the fixing constraints by specifying more negative values for the *-setup_margin* and *-hold_margin* options or by dropping those options entirely.

- *U - UPF restricts fixing the violation*

  Certain UPF cells cannot be sized or buffered. For details, see the "UPF Cells" section of this man page.

- *V - Net or cell is invalid or has dont_touch attribute*

  One or more leaf cells or nets under the cell have the *dont_touch* attribute set to *true*. Check to see if the reason is justified.

Upon completion of fixing, the command reports a summary of the remaining violations:

```
Remaining Violations:
  Violation Type                              Count
  ------------------------------------------------
  Total remaining violations                     10
  Unfixable violations                            8
```

This summary report shows 10 remaining violations, 8 of which are unfixable. You might be able to fix two more violations by running the *fix_eco_drc* command again. You can also try other options, such as relaxing the timing margins or increasing the alternative cell area limit.

To write out the unfixable violation report as a separate file, use the *-unfixable_reasons_prefix* option and specify a prefix for the file name. By default, it writes out two report files, one in text format and the other in CSV format.

The CSV report file is compatible with the PrimeTime GUI and spreadsheet programs. To read the report into the PrimeTime GUI, choose the Report > ECO Unfixed Violation menu command. You can use the PrimeTime GUI to investigate and fix the remaining violations. The entries in the violation table are cross-referenced to the timing reports, path inspector, and layout view shown in the GUI.

Writing Changes

After you fix violations using the *fix_eco_drc* command and other ECO commands, you can write out the design changes as a script by using the *write_changes* command. You can use the script to implement the same changes in another PrimeTime session or another tool (Design Compiler, IC Compiler, or IC Compiler II). Use the *-format* option of the *write_changes* command to specify the format of the script.

Multiple Fixing Scenarios With Fewer Hosts with Hybrid Timing View

In the hybrid timing view ECO flow, the tool uses a combination of live views for accuracy-critical ECO scenarios and static views of less critical scenarios. The static-view timing and violation information is merged into the live-view scenarios at intervals determined by the tool.

Suppose you have 50 scenarios for your fixing. In a regular DMSA analysis, you would need 50 worker processes simultaneously running on one or more machines.

In the hybrid timing view ECO flow, if you have 15 live-view scenarios that cover 90 percent of the violations, the tool can merge the remaining 35 scenarios with the 15 live-view scenarios as static views. Then, you can run fixing with only 15 DMSA worker processes instead of 50. Note that if the live views have a relatively low coverage of violations (for example, 80 percent), you might not be able to achieve the desired fixing coverage.

For more information, see the man page for the *fBwrite_eco_scenario_data* and the *fBstart_eco_scenarios* commands.

Physically Aware DRC Fixing

Before you run the *fix_eco_drc* command, use the *set_eco_options* command to specify the paths to the physical data, either IC Compiler II database files or LEF/DEF library and design data exchange files:

```
set_eco_options \\
  -physical_tech_lib_path LEF_tech_file_list \\
  -physical_lib_path LEF_lib_file_list \\
  -physical_design_path DEF_design_file_list

set_eco_options \\
  -physical_icc2_lib icc2_lib_directory_path \\
  -physical_icc2_blocks icc2_block_name_list
```

The *fix_eco_drc* command reads in the physical data files as specified by the *set_eco_options* command if they are not already read in and checked by the *check_eco* command. To view the chip layout in the GUI, open the GUI with the *gui_start* command and then choose Window > Layout Window.

For more information, see the man page for the *fix_eco_timing* command.

Fixing in Multiply Instantiated Modules (MIMs)

When you set the *eco_enable_mim* variable to *true* (the default is *false*), the tool performs the same ECO changes across each set of MIMs. In physically aware fixing, the tool recognizes a set of MIM instances when they share the same DEF file; in logic-only fixing, it recognizes them when they share the same parasitics file according to the *-path* option in the *read_parasitics* command.

For more information, see the man page for the *fix_eco_timing* command.

Per-Scenario ECO Margins

In distributed multi-scenario analysis, you can use different setup and hold margins for different scenarios by specifying their values in the *-drc_setup_margin* and *-drc_hold_margin* options of the *set_eco_options* command in the worker scripts.

Note that the *-setup_margin* and *-hold_margin* options of the *fix_eco_drc* command have priority over the corresponding options of the *set_eco_options* command.

Clock Network Fixing

When the *-cell_type* option is set to *clock_network*, the command makes changes in clock networks to fix violations. Both cell sizing and buffer insertion can be used.

In physically aware fixing, you must use the *set_eco_options* command with the *-physical_enable_clock_data* option before you use the *fix_eco_drc* command. You must specify valid buffer library cells that are available to the physical implementation tool, meeting any special rules such as nondefault routing (NDR) rules. The PrimeTime tool does not perform any extra rule checking for the provided buffers.

The PrimeTime tool uses all available alternative library cells to size cells in the clock network. If your design uses specialized library cells in the clock network, use a Tcl script similar to the following example to limit the sizing candidates to selected library cells. This example limits the library cells to those that have prefix "CLK" in their library cell base names.

```
define_user_attribute -type boolean -class lib_cell is_clk_cell
set clk_lcells [get_lib_cell */CLK*]
set_user_attribute $clk_lcells is_clk_cell true
set eco_alternative_cell_attribute_restrictions is_clk_cell
```

Examples

The following example fixes maximum transition violations using only the *size_cell* method.

```
pt_shell> fix_eco_drc -type max_transition -verbose \\
          -methods size_cell
 ...
Unfixable violations:
  A - There are available lib cells outside area limit
  C - The violation is in clock network
  I - Buffer insertion with given lib cells cannot fix the violation
  P - Driver cell of the violation is a port
  Q - Driver cell of the violation is a sequential cell
  S - Cell sizing with alternative lib cells cannot fix the violation
  T - Timing margin is too tight to fix the violation
  V - Driver cell of the violation is dont_touched
  U - UPF restricts fixing the violation
```

```
  Violation                                  Reasons
  -------------------------------------------------
  U1/I                                           P
  U6/I                                           S

Remaining Violations:
  Violation Type                               Count
  -------------------------------------------------
  Total remaining violations                       2
  Unfixable violations                             2
```

After fixing, two unfixable violations remain. The unfixable reason "S" shows that U6/I was not fixed because none of the alternative library cells can fix the maximum transition violation. The reason "P" shows that U1/I was not fixed because its driver is a port, which cannot be resized.

In the next run, you try the buffer insertion method to fix the violation using the three buffer library cells BUFX1, BUFX2, and BUFX3.

```
pt_shell> fix_eco_drc -type max_transition -verbose \\
          -methods insert_buffer -buffer_list { BUFX1 BUFX2 BUFX3 }

Unfixable violations:
  A - There are available lib cells outside area limit
  C - The violation is in clock network
  I - Buffer insertion with given lib cells cannot fix the violation
  P - Driver cell of the violation is a port
  Q - Driver cell of the violation is a sequential cell
  S - Cell sizing with alternative lib cells cannot fix the violation
  T - Timing margin is too tight to fix the violation
  V - Driver cell of the violation is dont_touched
  U - UPF restricts fixing the violation

  Violation                                  Reasons
  -------------------------------------------------
  U6/I                                           I

Remaining Violations:
  Violation Type                               Count
  -------------------------------------------------
  Total remaining violations                       1
  Unfixable violations                             1
```

The command fixed the first violation but could not fix the second one because the provided buffer library cells were not strong enough.

In the next run, you could try specifying a larger list of library buffer cells, and you could try using both the *size_cell* and *insert_buffer* methods for maximum flexibility.

The following example performs physically aware fixing.

```
pt_shell> fix_eco_drc -type max_transition -verbose \\
          -methods insert_buffer -buffer_list {BUFX1 BUFX2 BUFX3} \\
          -physical_mode open_site
 ...
Unfixable violations:
  A - There are available lib cells outside area limit
  C - The violation is in clock network
  D - Cell or net is located in high density area
  E - Physical information is incomplete or unavailable
  H - Logical and physical hierarchies are inconsistent
  I - Buffer insertion with given lib cells cannot fix the violation
  O - No open free site is available
  P - Driver cell of the violation is a port
  Q - Driver cell of the violation is a sequential cell
  R - No locations are available in parasitics or location transformation
 failed
  S - Cell sizing with alternative lib cells cannot fix the violation
  T - Timing margin is too tight to fix the violation
  V - Driver cell of the violation is dont_touched
  U - UPF restricts fixing the violation

  Violation                                  Reasons
  -------------------------------------------------
  U1/I                                             O

Remaining Violations:
  Violation Type                               Count
  -------------------------------------------------
  Total remaining violations                       1
  Unfixable violations                             1
```

The unfixable violation report lists some possible unfixable violation reasons (codes D, E, H, and O) related to physically aware fixing that are not shown in the logic-only unfixable reason report. The unfixable reason "O" shows that U1/I was not fixed because there is no empty free site available.

The following example shows a report of remaining violations when the *fix_eco_drc* command is run in distributed multi-scenario analysis (DMSA) mode. There are two scenarios, scen1 and scen2. There is one remaining violation in scen1 that is unfixable, but the remaining violation in scen2 could be fixed in the next run.

```
Remaining Violations:
  Scenario                   Total  Unfixable
  -----------------------------------------
  scen1                          1          1
  scen2                          1          0
```

The following example fixes noise violations using both the *size_cell* and *insert_buffer* methods.

```
pt_shell> fix_eco_drc -type noise -verbose \\
           -methods {size_cell insert_buffer} \\
```

```
            -buffer_list { BUFX1 BUFX2 BUFX3 } \\
            -physical_mode open_site
 ...
Unfixable violations:
  A - There are available lib cells outside area limit
  C - The violation is in clock network
  D - Cell or net is located in high density area
  E - Physical information is incomplete or unavailable
  H - Logical and physical hierarchies are inconsistent
  I - Buffer insertion with given lib cells cannot fix the violation
  O - No open free site is available
  P - Driver cell of the violation is a port
  Q - Driver cell of the violation is a sequential cell
  R - No locations are available in parasitics or location transformation
 failed
  S - Cell sizing with alternative lib cells cannot fix the violation
  T - Timing margin is too tight to fix the violation
  V - Driver cell of the violation is dont_touched
  U - UPF restricts fixing the violation

  Violation                                  Reasons
  ------------------------------------------------
  U1/I                                             O

Remaining Violations:
  Violation Type                               Count
  ------------------------------------------------
  Total remaining violations                       1
  Unfixable violations                             1
```

One unfixable violation remains. The unfixable reason "O" shows that U1/I was not fixed because there is no empty free site available.

The following example uses the *-insert_inverter_pair* method to perform maximum transition fixing in a clock network.

```
pt_shell> fix_eco_drc -type max_transition -cell_type clock_network \\
            -buffer_list {INVX1 INVX2 INVX4} \\
            -methods insert_inverter_pair
```

The following example uses the *-insert_inverter_pair* method to perform maximum transition fixing in data paths.

```
pt_shell> fix_eco_drc -type max_transition -cell_type combinational \\
            -buffer_list {INVX1 INVX2 INVX4} \\
            -methods insert_inverter_pair
```

The following example uses only the pin buffering method to perform maximum transition fixing.

Feedback

```
pt_shell> fix_eco_drc -max_transition \\
           -methods {insert_buffer_at_load_pins
 insert_buffer_at_driver_pins} \\
           -buffer_list {BUFX1 BUFX2 BUFX3}
```

The following example shows usage of the *-slack_lesser_than* option to fix large noise violations with slack worse than -0.05, thereby ignoring violations with slack range between -0.05 and 0.0.

```
pt_shell> fix_eco_drc -type noise -slack_lesser_than -0.05 \\
           -buffer_list {BUFX1 BUFX2 BUFX4}
```

The following example optimizes victim nets with delta delay cost of more than 0.05 time units and that belong to a path with negative timing slack, using both the *size_cell* and *insert_buffer* fixing methods.

```
pt_shell> fix_eco_drc -type delta_delay -verbose \\
  -delta_delay_threshold 0.05 \\
  -methods {size_cell insert_buffer} \\
  -buffer_list { BUFX1 BUFX2 BUFX3 } \\
  -physical_mode open_site
```

See Also

- estimate_eco
- fix_eco_timing
- report_analysis_coverage
- report_constraint
- report_eco_scenarios
- report_noise
- report_si_bottleneck
- report_cell_em_violation
- report_timing
- set_dont_use
- set_eco_options
- start_eco_scenarios
- write_changes
- write_eco_scenario_data

Feedback

- eco_allow_sizing_with_lib_cell_attributes
- eco_alternative_area_ratio_threshold
- timing_save_pin_arrival_and_required

---

fix_eco_leakage

Performs leakage recovery driven by signoff timing by replacing lower priority cells with higher priority cells.

Syntax

```
status fix_eco_leakage
   -pattern_priority pattern_list
   [-cell_type cell_types]
   [-attribute attribute_name]
   [-verbose]
   [-setup_margin margin]
```

Data Types

```
pattern_list       list
cell_types         list
attribute_name     string
margin             float
```

Arguments

`-pattern_priority` *pattern_list*

Specifies a list of library cell patterns in the order of highest to lowest priority.

`-cell_type` *cell_types*

Specifies a cell type for leakage recovery. Allowed values are *combinational* and *sequential*. The default is *{sequential combinational}*, which specifies that both sequential and combinational cells are swapped for leakage recovery.

`-attribute` *attribute_name*

Specifies an attribute name that can be used to match patterns specified by the *-pattern_priority* option. When you specify this option, PrimeTime uses the value of the attribute, instead of library cell names, for pattern matching.

`-verbose`

Shows additional information during leakage recovery process. During distributed multi-scenario analysis, the report includes setup violation information from each scenario.

`-setup_margin` *margin*

Specifies an additional timing margin to be applied to setup slacks during leakage recovery. By default, the setup margin is zero. If you specify a positive value, PrimeTime preserves the specified amount of slack during leakage recovery. If you specify a negative value, PrimeTime uses the specified amount of slack to swap more cells during leakage recovery.

Description

This command seeks to recover leakage (reduce leakage current) as much as possible by swapping cells that have positive setup slacks, which can be reduced, without creating new timing violations. You can use the *fix_eco_leakage* command in the multicore analysis and distributed multi-scenario analysis flows within PrimeTime.

Run the *fix_eco_leakage* command in distributed multi-scenario analysis flow to consider signoff timings from all scenarios. PrimeTime analyzes timing across all scenarios during the leakage recovery process and maintains signoff-state timing in all scenarios.

You must specify the required *-pattern_priority* option with a *pattern_list*. This option determines the priority order that PrimeTime follows when replacing lower priority cells with higher priority cells. Therefore, the first pattern indicates the cells with the lowest leakage (highest priority for swapping), and the last pattern indicates the cells with the highest leakage (lowest priority for swapping).

PrimeTime swaps cells if the following conditions are met:

- The original library cell has enough positive setup slack.
- The new library cell is compatible with the original library cell and satisfies the pattern matching criteria (see "Pattern Matching and Priority Order").
- The new library cell has higher priority than the original library cell.
- The new library cell does not create new setup, max_transition, or max_capacitance violations.

Pattern Matching and Priority Order

To determine the priority of cell swapping, the *fix_eco_leakage* command can perform pattern matching if the library cell names have a changing pattern and a fixed pattern. A changing pattern is a text string that represents different leakage values, such as HVT, NVT, and LVT. A fixed pattern is a text string that does not change with different leakage values, such as BUF1X.

To apply the pattern matching algorithm and identify the candidates for cell swapping, use *-pattern_priority* option; specify the cells in the *pattern_list* in the order of highest to lowest priority (most to least preferred).

For example, suppose you have three library cells: HVT_BUF1X, NVT_BUF1X, and LVT_BUF1X. They are the same buffers except with different leakage values. The HVT_BUF1X library cell has the lowest leakage, which is most preferred; the LVT_BUF1X library cell has the highest leakage, which is least preferred.

In this example, you would specify the *-pattern_priority* option with the following *pattern_list*: {HVT NVT LVT}. PrimeTime swaps cells in the following order:

- LVT_BUF1X can be replaced by HVT_BUF1X or NVT_BUF1X.
- NVT_BUF1X can be replaced by HVT_BUF1X.
- HVT_BUF1X cannot be replaced; it has the highest priority.

In this case, HVT_BUF1X has highest priority as the most preferred cell. When the tool finds an LVT_BUF1X cell, it tries to replace it with HVT_BUF1X, if doing so would not create any new timing violations. If not, then the tool tries to replace LVT_BUF1X with NVT_BUF1X. The tools also tries to replace any NVT_BUF1X cell with HVT_BUF1X. However, the tool does not change HVT_BUF1X because it is the most preferred cell.

Specifying the Pattern List

Pattern matching supports prefix, mid-fix, and postfix. For example, you can specify {HVT NVT LVT} for the following library cell names.

```
Prefix : HVT_BUF1X,  LVT_BUF1X,  NVT_BUF1X
  Mid-fix: BUF_HVT_1X, BUF_NVT_1X, BUF_LVT_1X
  Postfix: BUF1X_HVT,  BUF1X_NVT , BUF1X_LVT
```

If multiple patterns overlap each other, the largest pattern is chosen. For example, if you specify {LVT LLVT} for the pattern, PrimeTime replaces BUF1X_LLVT (with matching LLVT pattern) by BUF1X_LVT (with matching LVT pattern), even though the LVT pattern also matches BUF1X_LLVT.

Swapping Cells By Attributes Instead of Cell Names

Even if your libraries do not have the naming convention described previously, you can still recover leakage by using a user-defined attribute and the *fix_eco_leakage -attribute* command. This also gives you great flexibility to customize your own flow and apply different techniques in different stages of your design.

For example, suppose that there are six library cells: BUF_A, B_BUF, BF1X, INV_A, B_INV, and IV2X. Although the pattern matching cannot be used with these cell names, you can still recover leakage by using user-defined attributes. In this example, BUF_A and INV_A are the most preferred cells, and BF1X and IV2X are the least preferred cells. To perform cell swapping with attributes, use the following commands:

```
# Define a user-defined attribute 'eco_pattern" for lib cell.
# Assign different value to each library cell with "good ok bad"
#
```

```
define_user_attribute eco_pattern -type string -class lib_cell
set_user_attribute -class lib_cell [get_lib_cell BUF_A] eco_pattern
 "buf_good"
set_user_attribute -class lib_cell [get_lib_cell B_BUF] eco_pattern
 "buf_ok"
set_user_attribute -class lib_cell [get_lib_cell BF1X ] eco_pattern
 "buf_bad"
set_user_attribute -class lib_cell [get_lib_cell INV_A] eco_pattern
 "inv_good"
set_user_attribute -class lib_cell [get_lib_cell B_INV] eco_pattern
 "inv_ok"
set_user_attribute -class lib_cell [get_lib_cell IV2X ] eco_pattern
 "inv_bad"
#
# Specify pattern priority using an attribute
#
fix_eco_leakage -pattern_priority {good ok bad} -attribute eco_pattern
```

Note that the name of user-defined attribute for each library cell must be unique. The name of user-defined attribute for a group of library cells that can be swapped to one another must have a common base name (for example, buf or inv) along with a name to specify priority (for example, good, ok, or bad). This is required for PrimeTime to know which cells can be swapped relative to one another.

Cells That Are Not Swapped

The recovery process does not swap the following cells:

- Cells with *dont_touch*, *dont_use*, or *pt_dont_use* attributes

  As in Design Compiler and IC Compiler, a *dont_touch* attribute applied to an upper-level hierarchical cell propagates downward into the hierarchy. If a different *dont_touch* value is applied deeper in the hierarchy, it takes precedence over any higher-level *dont_touch* attributes.

  To prevent the use of specific library cells for leakage recovery, apply the *dont_use* attribute by using the *set_dont_use* command. Alternatively, you can also apply the *pt_dont_use* user-defined attribute.

  In distributed multi-scenario analysis (DMSA) fixing, a restriction in one scenario also should be applied to other scenarios. You should apply the *dont_touch* and *dont_use* attributes to all scenarios to prevent design changes across all scenarios for the affected cells and nets.

- Cells in the clock network
- Cells in the fanin cone of transparent latches

  By default, the cells in the fanin cone of transparent latches are not swapped because it is difficult to recover timing violations created from swapping. The default latch

analysis always sees zero slack at borrowing latches due to the single-segment nature of timing paths.

To allow swapping cell in the fanin cone of transparent latches, enable advanced latch analysis by setting the *timing_enable_through_paths* variable to *true*. Advanced latch analysis has global visibility of timing paths through latches and can see whether a timing violation found at the downstream of the latch is caused by swapping cells in the upstream.

Unconstrained Cells

In distributed multi-scenario analysis, a cell is unconstrained if the cell is not constrained in all scenarios. If a cell is not constrained in one scenario but constrained in another scenario, the cell is constrained in distributed multi-scenario analysis.

You can control swapping unconstrained cells by specifying the *eco_leakage_exclude_unconstrained_cells* variable. If the variable is set to *false*, PrimeTime swaps unconstrained cells. If the variable is set to *true*, PrimeTime does not swap unconstrained cells.

UPF

If a cell has the *always_on*, *is_isolation*, *is_retention*, or *is_level_shifter* low-power control attribute set to *true*, it is not considered for cell swapping.

Honoring Footprint Attributes in the Library

If you want PrimeTime to honor a footprint attribute from libraries, you need to import the attribute and specify the attribute name in the *eco_alternative_cell_attribute_restrictions* variable. Suppose your library has the *cell_footprint* attribute to indicate cell footprint. Then, the following commands restrict the *fix_eco_leakage* command to swap only footprint-compatible cells.

```
# Import 'cell_footprint' attribute to PrimeTime
define_user_attribute cell_footprint -class lib_cell -import -type string

# Restrict compatible cells only to the same cell_footprint
set eco_alternative_cell_attribute_restrictions {cell_footprint}

# Swappable cells should satisfy both the pattern priority and
# the cell_footprint attribute value
fix_eco_leakage -pattern_priority {HVT NVT LVT}
```

Examples

The following example shows how to recover leakage with the library cells having prefix HVT, NVT, and LVT for their names. Both sequential and combinational cells are swapped.

```
pt_shell> fix_eco_leakage -pattern_priority {HVT NVT LVT}
```

The following example shows how to swap only combinational cells.

Feedback

```
pt_shell> fix_eco_leakage -cell_type combinational -pattern_priority {HVT
 NVT LVT}
```

Here is typical script example for leakage recovery in distributed multi-scenario analysis.

```
#-------------------------------------------------------------------------
# Update timing & global reporting in manager process
#-------------------------------------------------------------------------
remote_execute {
  set timing_save_pin_arrival_and_slack true; update_timing -full
}
set pattern "HVT NVT LVT"
report_global_timing
report_constraint -all -max_capacitance -max_transition

#-------------------------------------------------------------------------
# Before leakage recovery:
#   Report timing, power, cell usages for each worker
#-------------------------------------------------------------------------
remote_execute {
  set pattern "HVT NVT LVT"
  report_global_timing
  report_constraint -all -max_capacitance -max_transition
  set power_enable_analysis true
  report_power -group {...} -threshold_voltage_group -pattern $pattern
}

#-------------------------------------------------------------------------
# Before leakage recovery:
#   Check cell swap compatibility. Detect any misuse of patterns
#   by using the -alternative_lib_cells and -show_others
#   options.
#-------------------------------------------------------------------------
remote_exec {
  report_cell_usage -pattern $pattern -alternative_lib_cells -show_others
}

#-------------------------------------------------------------------------
# Execute signoff-driven leakage recovery.
#-------------------------------------------------------------------------
fix_eco_leakage -pattern $pattern -verbose

#-------------------------------------------------------------------------
# Reporting after leakage recovery
#-------------------------------------------------------------------------
report_global_timing
report_constraint -all -max_capacitance -max_transition
remote_execute {
  report_global_timing
  report_constraint -all -max_capacitance -max_transition
  set power_enable_analysis true max_transition
  report_power -group {...} -threshold_voltage_group -pattern $pattern
```

```
}
exit
```

See Also

- fix_eco_drc
- fix_eco_timing
- report_cell_usage
- set_dont_use
- eco_alternative_area_ratio_threshold
- timing_save_pin_arrival_and_slack

---

fix_eco_power

Performs power recovery by downsizing or swapping cells in paths with positive setup slack or by removing buffers.

Syntax

```
status fix_eco_power
   [-cell_type cell_types]
   [-setup_margin margin]
   [-hold_margin margin]
   [-verbose]
   [-power_attribute power_attribute_name]
   [-pba_mode none | path | exhaustive | ml_exhaustive]
   [-pba_path_selection_options option_string]
   [-pattern_priority pattern_list]
   [-attribute attribute_name]
   [-methods method_list]
   [-current_library]
   [-power_mode total | dynamic | leakage]
   [-leakage_scenario scenario_name]
   [-dynamic_scenario scenario_name]
   [-training_data file_list]
   [-timeout seconds]
   [-start_end_type reg_to_reg]
```

Data Types

```
cell_types            list
margin                float
power_attribute_name  string
option_string         string
pattern_list          list
attribute_name        string
method_list           list
```

```
scenario_name          string
file_list              list
```

Arguments

`-cell_type` *cell_types*

Specifies the types of cells considered for downsizing or swapping, either *combinational*, *sequential*, or *clock_network*. Cell types *combinational* and *sequential* can be specified together, and the default is both, *-cell_type {combinational sequential}*; cell type *clock_network* cannot be specified together with other cell types. Only sequential cells that synchronous (flip-flops, not transparent latches) can be modified during power recovery. Note that users need to specify *-power_mode dynamic* and *-dynamic_scenario* options together with the cell type *clock_network* to use PrimePower power analysis data to drive clock dynamic power recovery.

Note: The *clock_network* value requires a PrimeECO license.

`-setup_margin` *margin*

Specifies the amount of setup timing margin to keep during power recovery, in library time units. For example, setting the margin to 0.2 requires the power recovery process to keep at least 0.2 of positive setup slack in each path. Specifying a negative margin allows timing violations up to the specified negative slack value. The default margin is zero.

During power recovery, if some path endpoints have a setup slack within the specified margin, the log file reports them as "endpoints with setup slack lesser than margin." These endpoints are not violations with negative slack (less than zero slack), but rather these endpoints fall within a slack range which power recovery will try to maintain the current setup slack relative to these endpoints.

`-hold_margin` *margin*

Specifies the amount of hold timing margin to keep during buffer removal or clock dynamic power recovery, in library time units. For example, setting the margin to 0.1 requires the buffer removal or clock dynamic power recovery process to keep at least 0.1 of positive hold slack in each path. The default margin is zero. This option can be used only when the *-methods* option is set to *remove_buffer* or when the *-cell_type* option is set to *clock_network*.

`-verbose`

Shows additional information during the power recovery process. During distributed multi-scenario analysis, the report includes setup violation information from each scenario.

The verbose mode also generates an "Unusable Cells" report upon completion of power recovery process if the *eco_power_report_max_unusable_cells*

variable is set to a positive integer. The report shows a list of cells where power recovery was considered, the reason that the power recovery was not performed at the cell. For details, see "Unusable Cell Report" in the DESCRIPTION section of this man page.

`-power_attribute` *power_attribute_name*

Specifies an attribute for prioritizing library cells used for downsizing or swapping. When you use this option, the tool uses the value of the power attribute, instead of library cell's area attribute, to prioritize usage of library cells for downsizing or swapping. Library cells with lower power attribute values are preferred over those with higher values.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Specifies one of the following path-based timing analysis modes:

- *none* (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- *path* - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths.
- *exhaustive* - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode.
- *ml_exhaustive* - Performs machine learning based exhaustive path-based analysis. This mechanism will deploy machine learning techniques to trade runtime vs accuracy during the design flow. Accuracy and runtime will match the *-pba_mode exhaustive* option as the design approaches signoff.

This option works like the *-pba_mode* option in the *report_timing* and *get_timing_paths* commands. The power recovery process uses the specified analysis mode to make sure that no new timing violations are introduced, or that existing timing violations do not become worse. Path-based analysis is more accurate (less pessimistic) than conventional graph-based analysis, so it reports more positive slack and allows more opportunities for power recovery at the cost of more runtime and memory. For more information, see "Path-Based Timing Analysis During Power Recovery" in the DESCRIPTION section.

`-pba_path_selection_options` *option_string*

Specifies the path selection options for the *-pba_mode path* option. The command uses these options to collect timing paths and determine the timing

slack available in these paths for power recovery. Only the following options are supported in the option string:

- *-nworst* - Specifies the maximum number of worst paths per endpoint considered for path-based analysis. The default is 1.
- *-max_paths* - Specifies the maximum total number of paths considered for path-based analysis. The default is to automatically choose a value large enough to cover all violating endpoints.
- *-cover_design* - Covers the entire design by including the worst path through each violating pin in the design during path-based analysis. This path selection option can be used only by itself, without the other two options.

If you use one or both of the *-nworst* and *-max_paths* options, be sure to specify values large enough to cover the violating endpoints that you want to be fixed. For more information, see "Path-Based Power Recovery" in the DESCRIPTION section.

`-timeout` *`seconds`*

Specifies a timeout limit, in seconds, that sets a maximum total runtime for the command. At the end of each iteration, the tool checks the total elapsed wall clock time. If the time limit is reached, the command stops running and ends with the current iteration. You cannot use this option with *-pattern_priority* and *-method remove_buffer*.

`-pattern_priority` *`pattern_list`*

Specifies a list of library cell patterns (either name prefixes or attribute strings) in order, from highest to lowest priority. When the command performs cell swapping, it uses a library cell with a name prefix or attribute setting that matches an earlier entry in the pattern list to replace an existing cell that matches a later entry in the list.

`-attribute` *`attribute_name`*

Specifies the library cell attribute that is considered by the *-pattern_priority* option. When you use the *-attribute* option, the command gives priority to a library cell with the specified attribute matching an earlier entry in the *-pattern_priority* list; it replaces an existing cell with the attribute matching a later entry in the list.

If you omit the *-attribute* option, the command considers the library cell name instead of a library cell attribute.

`-methods` *`method_list`*

Specifies the power recovery method, either *size_cell* or *remove_buffer*, but not both. The default is *size_cell*. The *size_cell* method downsizes or swaps cells

in paths with positive setup slack. The *remove_buffer* method removes buffers without introducing or worsening timing and DRC violations.

The *remove_buffer* method cannot be used with any of the following options: *-cell_type*, *-pattern_priority*, *-attribute*, *-power_attribute*, *-current_library*, and *-power_mode*.

`-current_library`

Specifies that during cell sizing or swapping using the *-power_attribute* or *-power_mode* option, the replacement cell must come from the same library as the original cell. By default, the command looks for replacement cells in all libraries, except for downsizing based on cell area, when the command always looks for cells only in the same library.

Using the *-current_library* option can improve runtime because fewer cells are considered as potential replacements. However, it might degrade quality of results because the variety of available alternative library cells could be reduced. You cannot use this option with the *-pattern_priority* option.

`-power_mode total | dynamic | leakage`

Specifies usage of PrimePower power analysis data, and specifies the types of power considered during power recovery: *dynamic* power only, *leakage* power only, or *total* power (both dynamic and leakage). When you use this option, PrimePower power analysis data must be available from previous usage of the *update_power* or *report_power* command. You cannot use this option with the *-power_attribute*, *-pattern_priority*, or *-methods remove_buffer* option.

If you do not use the *-power_mode* option, power recovery does not use PrimePower power analysis data. Instead, the command chooses replacement cells based on the cell area by default, or based on a cell power attribute or string pattern list if you use the *-power_attribute* or *-pattern_priority* option.

`-leakage_scenario` *`scenario_name`*

Specifies the name of the scenario from which to get the leakage power data in a DMSA flow when using the *-power_mode* option. It should be the scenario with the worst leakage power.

`-dynamic_scenario` *`scenario_name`*

Specifies the name of the scenario from which to get the dynamic power data in a DMSA flow when using the *-power_mode* option. It should be the scenario with the worst dynamic power.

The *-leakage_scenario* and *-dynamic_scenario* options can be set to the same scenario or two different scenarios.

`-training_data` *`file_list`*

Specifies a list of existing files that contain training data, used for faster power optimization in the current session. When you use this option, power optimization uses only the listed files and does not use the *training_data_directory* variable to read existing training data. (Note that the command still considers the *training_data_directory* setting for writing out training data learned in the current session for usage in future sessions.)

By default, without the *-training_data* option, if the *training_data_directory* varible is set to a directory name, the command uses all training data files in that directory for faster power optimization in the current session, and writes new training data to that directory upon completion of power optimization. Otherwise, if the *training_data_directory* variable is set to an empty string (the default variable setting), ECO power optimization does not use training data in the current session and does not learn or save training data for future sessions.

The argument of the *-training_data* option is a list of files. You can specify the files using absolute or relative paths. If you use relative paths, the command looks for the training data files in the current working directory.

Training data files are generated by earlier usage of the *fix_eco_power* command with the *training_data_directory* variable set to the name of a directory, or by usage of the *record_training_data*, *size_cell*, and *write_training_data* commands. To use training data generated by both methods, put all the training data files in a single directory and set the *training_data_directory* variable to the name of that directory.

`-start_end_type reg_to_reg`

Specifies the tool to optimize cells only on register-to-register paths without degrading interface timing. When the *start_end_type* option is specified, cells that are reachable from input ports (excluding clock input ports) and output ports are not optimized, and cells in register-to-register paths are optimized using the register-to-register timing slack. This option cannot be specified without *-pba_mode* option, and cannot be specified with *cell_type clock_network_cell*.

Description

The *fix_eco_power* command tries to recover power and area as much as possible by downsizing cells in paths with positive setup slack or by removing buffers without introducing or worsening timing and DRC violations.

After power recovery, you can write out the design changes as a script by using the *write_changes* command. You can use the script to implement the same changes in another PrimeTime session or another tool such as IC Compiler II. A list of design changes created in this flow is called an engineering change order (ECO).

By default, the *fix_eco_power* command performs power and area recovery by downsizing cells in all paths with positive setup slack. The command options let you specify any one of the following power recovery methods:

- Replace cells to minimize area (the default)
- Replace cells to minimize a library cell numeric attribute (use the *-power_attribute* option)
- Replace cells based on library cell preference, as specified by an explicit string priority list (use the *-pattern_priority* option)
- Replace cells to minimize switching, leakage, or total power using power analysis data generated by the *update_power* or *report_power* command (use the *-power_mode* option)
- Remove buffers (use the *-methods remove_buffer* option)

You can apply multiple fixing methods by running the *fix_eco_power* command repeatedly with different option settings.

For all of the cell replacement methods, replacement occurs only when the following conditions are met:

- The change does not introduce or worsen any timing violations or DRC violations (max_transition or max_capacitance).
- The replacement cell has the same logical function as the original cell and meets any usage restrictions defined by the *eco_alternative_cell_attribute_restrictions* variable.
- The replacement cell is preferred over the original cell in one of the following ways, depending on the option settings: less area, smaller power attribute value, higher-priority name or attribute string, or less power based on PrimePower power analysis data.

Power recovery is an iterative process. The *fix_eco_power* command starts by making the targeted changes and then analyzes the design for new timing and DRC violations (or worsening of existing violations) caused by the changes. It incrementally backs out of previous changes to meet the timing and DRC constraints, and also explores further changes for improved power recovery, a process called "tuning." It analyzes the design again and repeats this process until it determines that the cost of further progress exceeds the potential benefit.

You can use path-based timing analysis during power recovery to reduce analysis pessimism and give more timing slack, allowing more opportunities for power recovery, at the cost of more runtime. To invoke this option in the *fix_eco_power* command, set the *-pba_mode* option to *path* or *exhaustive*.

The *fix_eco_power* command is compatible with single-core analysis, distributed multi-scenario analysis (DMSA), and multicore analysis. To ensure consistent results during DMSA power recovery, apply any *dont_touch*, *dont_use* and *size_only* attributes identically to all scenarios.

Replacing Cells to Minimize Area (Default Method)

By default, the *fix_eco_power* command by itself (without options) performs power and area recovery by downsizing cells in all paths with positive setup slack. For each cell in these paths, the command looks for alternative cells in the same library that have the same logical function but a smaller area, and considers using them to replace the existing cell. For two library cells with the same area, the one with better timing is preferred.

A smaller cell typically uses less power and has less drive strength, so the tool considers both the area reduction and timing effects of potential replacement cells along each timing path.

You might want to specify which replacement cells are allowed be used for downsizing certain cells. For example, if a library contains both low-Vt and high-Vt cells, you might want high-Vt cells to be replaced only by other high-Vt cells, because a smaller low-Vt cell would reduce switching power but increase leakage power. To do this, assign the library cells into groups and set the *eco_alternative_cell_attribute_restrictions* variable as shown in the following example.

```
# Define a lib_cell attribute called "eco_group"
define_user_attribute eco_group -type string -class lib_cell

# Assign library cells to named groups
set_user_attribute -class lib_cell [get_lib_cell INVX1_LVT] eco_group
 "INV_L"
set_user_attribute -class lib_cell [get_lib_cell INVX2_LVT] eco_group
 "INV_L"
set_user_attribute -class lib_cell [get_lib_cell INVX1_HVT] eco_group
 "INV_H"
set_user_attribute -class lib_cell [get_lib_cell INVX2_HVT] eco_group
 "INV_H"
set_user_attribute -class lib_cell [get_lib_cell BUFX1_LVT] eco_group
 "BUF_L"
set_user_attribute -class lib_cell [get_lib_cell BUFX2_LVT] eco_group
 "BUF_L"
set_user_attribute -class lib_cell [get_lib_cell BUFX1_HVT] eco_group
 "BUF_H"
set_user_attribute -class lib_cell [get_lib_cell BUFX2_HVT] eco_group
 "BUF_H"

# Restrict compatible replacement cells to the same group
set_app_var eco_alternative_cell_attribute_restrictions {eco_group}

# Perform cell downsizing with restricted cell replacement
fix_eco_power
```

Downsizing based on cell area always uses replacement cells from the same library as the original cell, and replacement cells are further restricted to the same group when you set the *eco_alternative_cell_attribute_restrictions* variable.

The other power recovery methods use cells from all available libraries by default. To restrict the selection to only cells from the same library as the original cell when using the *-power_attribute* or *-power_mode* option, also use the *-current_library* option.

Replacing Cells Based on Name Substring Pattern Priority

To perform power recovery by replacing cells in all paths with positive setup slack, using a library cell name substring to determine the order of preference, use the *-pattern_priority* option. The tool attempts to replace each cell that has a lower-priority substring with a cell that has a higher-priority substring, without introducing timing or DRC violations.

For example, suppose that you have three buffer library cells with the same area but different threshold voltages and delays, named HVT_BUF1X, MVT_BUF1X, and LVT_BUF1X (for high-Vt, mid-Vt, and low-Vt). For power recovery, the cells are preferred in that order. To perform power recovery based on these name prefixes, use the following command:

```
fix_eco_power -pattern_priority {HVT MVT LVT}
```

For each cell that contains "LVT" in the library cell name, the command looks for another cell with a similar name, with "HVT" or "MVT" in place of "LVT", and performs the replacement using that cell if it can. In this example, it replaces instances of LVT_BUF1X with HVT_BUF1X if no violations are introduced, or with MVT_BUF1X otherwise, if that replacement does not introduce violations. Instances of HVT_BUF1X are never replaced because they already have the highest priority.

The varying substrings {HVT MVT LVT} can be anywhere in the library cell name: at the beginning (prefix), inside, or at the end (suffix). For example, the substring set {HVT MVT LVT} works with the following sets of library cell names:

```
Prefix: HVT_BUF1X, MVT_BUF1X, LVT_BUF1X
  Inside: BUF_HVT1X, BUF_MVT1X, BUF_LVT1X
  Suffix: BUF1X_HVT, BUF1X_MVT, BUF1X_LVT
```

This method considers the cells that differ by the substring part of the name but otherwise exactly match by name.

Replacing Cells Based on Library Cell Attribute Pattern Priority

To perform power recovery by replacing cells in all paths with positive setup slack, using a library cell attribute substring to determine the order of preference, use the *-pattern_priority* option to specify the string priority list, together with the *-attribute* option to specify the library cell attribute name. The tool attempts to replace each cell that has a lower-priority attribute substring with a cell that has a higher-priority attribute substring, without introducing timing or DRC violations.

When you use the *-attribute* option with the *-pattern_priority* option, the tool finds replacement cells by looking at a library cell attribute instead of the library cell name. This method gives you the flexibility to customize your own cell replacement flow and apply different techniques at different stages of the design.

For example, suppose that you have library cells named BUF_A, B_BUF, BF1X, INV_A, B_INV, and IV2X. In this fixing flow, BUF_A and INV_A are preferred the most, and BF1X and IV2X are preferred the least. To define the preference attribute and perform cell replacement optimization, use the following commands:

```
# Define a string attribute named eco_pattern for lib_cell
# Assign different value to each library cell with string "good ok bad"
define_user_attribute eco_pattern -type string -class lib_cell
set_user_attribute -class lib_cell [get_lib_cell BUF_A] eco_pattern
 "buf_good"
set_user_attribute -class lib_cell [get_lib_cell B_BUF] eco_pattern
 "buf_ok"
set_user_attribute -class lib_cell [get_lib_cell BF1X ] eco_pattern
 "buf_bad"
set_user_attribute -class lib_cell [get_lib_cell INV_A] eco_pattern
 "inv_good"
set_user_attribute -class lib_cell [get_lib_cell B_INV] eco_pattern
 "inv_ok"
set_user_attribute -class lib_cell [get_lib_cell IV2X ] eco_pattern
 "inv_bad"

# Specify pattern priority using an attribute
fix_eco_power -pattern_priority {good ok bad} -attribute eco_pattern
```

For each cell that contains "bad" in the library cell's eco_pattern attribute, the command looks for another cell with a similar eco_pattern attribute setting, with "good" or "ok" in place of "bad", and performs the replacement using that cell if it can. In this example, it replaces instances of BF1X (buf_bad) with BUF_A (buf_good) if no violations are introduced, or with B_BUF (buf_ok) otherwise, if that replacement does not introduce violations. Instances of BUF_A (buf_good) are never replaced because they already have the highest priority.

This method considers the cells that differ by the substring part of the attribute string, with the rest of the attribute string matching exactly, and the replacement cell must be functionally equivalent.

In distributed multi-scenario analysis (DMSA), you must use the same string attribute settings for the same respective library cells across all scenarios. For example,

```
remote_execute {
  define_user_attribute eco_pattern -type string -class lib_cell
  set_user_attribute -class lib_cell [get_lib_cell BUF_A] eco_pattern
 "buf_good"
  set_user_attribute -class lib_cell [get_lib_cell B_BUF] eco_pattern
 "buf_ok"
  ...
```

```
}
fix_eco_power -pattern_priority {good ok bad} -attribute eco_pattern
```

Replacing Cells to Minimize a Numeric Power Attribute

To perform power recovery by replacing cells in all paths with positive setup slack, using a library cell attribute to determine the order of preference, use the *-power_attribute* option. The tool replaces cells to minimize the value of the specified library cell attribute without increasing the cell area. It can choose cells from all libraries, not just the library of the existing cell. The attribute can represent any type of quantity that you want to minimize, such as leakage power or switching power.

For example, suppose there are inverters named INV1X_LVT, INV2X_LVT, INV1X_HVT, and INV2X_HVT, contained in two libraries named LIB_L and LIB_H. You define a numeric library cell attribute named prw_attr and set the value of this attribute for the library cells as follows:

```
define_user_attribute pwr_attr -type float -class lib_cell
set_user_attr -class lib_cell [get_lib_cell LIB_H/INV1X_HVT] pwr_attr 1.0
set_user_attr -class lib_cell [get_lib_cell LIB_L/INV1X_LVT] pwr_attr 1.5
set_user_attr -class lib_cell [get_lib_cell LIB_H/INV2X_HVT] pwr_attr 2.0
set_user_attr -class lib_cell [get_lib_cell LIB_L/INV2X_LVT] pwr_attr 3.0
```

To perform power recovery and replace cells to minimize the pwr_attr value, use the following command:

```
fix_eco_power -power_attribute pwr_attr
```

The replacement cell must have a smaller value for the specified attribute and must have the same or smaller area. Only the library cells that have the specified attribute defined on them are considered as potential replacement cells.

In distributed multi-scenario analysis (DMSA), you must use the same value of power attribute for the same respective library cells across all scenarios. For example,

```
remote_execute {
  define_user_attribute pwr_attr -type float -class lib_cell
  set_user_attr -class lib_cell [get_lib_cell LIB_H/INV1X_HVT] pwr_attr
 1.0
  set_user_attr -class lib_cell [get_lib_cell LIB_L/INV1X_LVT] pwr_attr
 1.5
  set_user_attr -class lib_cell [get_lib_cell LIB_H/INV2X_HVT] pwr_attr
 2.0
  set_user_attr -class lib_cell [get_lib_cell LIB_L/INV2X_LVT] pwr_attr
 3.0
}
fix_eco_power -power_attribute pwr_attr
```

Feedback

Power Recovery Using PrimePower Analysis Data

The PrimePower tool performs comprehensive power analysis using actual switching activity and library-defined power data such as dynamic and leakage power of each cell. The tool performs a power analysis when you use the *update_power* or *report_power* command at the pt_shell prompt.

To use PrimePower power analysis data in power recovery, use the *fix_eco_power* command with the *-power_mode* option set to *dynamic*, *leakage*, or *total*. Then the power recovery process modifies the design to minimize the dynamic (switching) power, leakage power, or total (dynamic plus leakage) power as measured by the *update_power* command.

With the *-power_mode total* option, the tool chooses actions to reduce the total power the most, which can vary with local conditions in the design. For example, where not much timing slack is available, it can choose to recover power by either downsizing the cell or by increasing the cell threshold voltage, the choice depending on the local switching activity. High switching activity favors downsizing (to reduce switching power), whereas low switching activity favors increasing the threshold voltage (to reduce leakage). Where plenty of timing slack is available, it can do both.

In distributed multi-scenario analysis (DMSA) flow, the tool gets its dynamic power data from exactly one scenario, which you specify with the *-dynamic_scenario* option. Similarly, it gets its leakage power data from exactly one scenario, which you specify with the *-leakage_scenario* option. These options are used only in a DMSA flow. In general, you should specify the scenario showing the worst dynamic power and worst leakage power, respectively, for these two options. They could be the same scenario or two different scenarios.

By default, the PrimePower tool accounts for the internal power of a register cell's clock input pin as a member of the clock network, not as part of the register cell. You can change this behavior by setting the *power_clock_network_include_register_clock_pin_power* variable to *false*, so that power recovery is better reflected as part of the data path.

The following script performs total power recovery in a single-scenario flow.

```
restore_session Session1
# Enable PrimePower and set up power analysis
set_app_var power_enable_analysis true
set_app_var power_clock_network_include_register_clock_pin_power false
read_saif activity_file.saif
# or
# read_vcd activity_file.vcd
 ...
# Perform power analysis
update_power
# Report power before power recovery
report_power
```

```
fix_eco_power -power_mode total

# Report power improvement
report_power
```

The following script performs total power recovery in a DMSA flow with four scenarios. Scenario S1 has the worst leakage power and scenario S3 has the worst dynamic power.

```
current_session {S1 S2 S3 S4}
# update_power is only needed for power scenarios S1 and S3
current_scenario {S1 S3}
remote_execcute {
  # Enable PrimePower and set up power analysis
  set_app_var power_enable_analysis true
  set_app_var power_clock_network_include_register_clock_pin_power false
  read_saif activity_file.saif
  # or
  # read_vcd activity_file.vcd
  ...
  update_power
}
# Report power in power scenarios before power recovery
remote_execute -verbose { report_power }
current_scenario -all

fix_eco_power -power_mode total \\
 -leakage_scenario S1 \\
 -dynamic_scenario S3 \\

# Report power improvement in power scenarios
current_scenario {S1 S3}
remote_execute -verbose { report_power }
```

Removing Buffers

To remove buffers to recover power or area, use the *-methods {remove_buffer}* option of the *fix_eco_power* command. Using this method, the tool considers a buffer for removal when all the following conditions are met:

- The buffer is constrained for both hold and setup timing when the *eco_power_exclude_unconstrained_cells* variable is set to *true*. In a DMSA flow, it must be constrained in at least one scenario for hold timing and in at least one scenario for setup timing. When the *eco_power_exclude_unconstrained_cells* variable is set to *false*, tool also considers unconstrained buffers for removal. In a DMSA flow, a buffer must be unconstrained in all scenarios for both hold and setup timing to be considered unconstrained. If a buffer is constrained only for hold or setup timing, it will not be considered for removal regardless of the value set to *eco_power_exclude_unconstrained_cells* variable.

- Removing the buffer does not introduce any new hold or setup timing violation, or does not worsen existing violations, or does not cause the slack to become worse than the margins set by the *-hold_margin* or *-setup_margin* options.
- Removing the buffer does not introduce any new DRC violations: max_transition, max_capacitance, or max_fanout.

The following buffers are *not* considered for removal:

- A buffer in a clock network
- A buffer with one or more timing exceptions defined on itself or its pins i.e. defined through *set_false_path, set_sense, and/or set_data_check commands.*
- A buffer with its *dont_touch* attribute set to true
- A buffer with its *size_only* attribute set to true
- A buffer with its *always_on* attribute set to true
- A buffer in the fanin cone of a transparent latch (default restriction)

To allow buffer removal in the fanin cone of a transparent latch, enable advanced latch analysis by setting the *timing_enable_through_paths* variable to *true*.

When the tool removes a buffer, it stitches the parasitics at the buffer input and output pins together to form a new combined parasitic network, which is seen by the driver of the removed buffer. This behavior is based on the assumption that the physical implementation tool will maintain the existing routing and replace the buffer with a short-circuit connection.

This *fix_eco_power -methods {remove_buffer}* command only removes buffers. To also perform cell sizing or swapping for power recovery, use a separate *fix_eco_power* command with appropriate option settings.

Cells That Are Not Modified

By default, the power recovery methods do not downsize or swap the following cells:

- Cells in clock networks when cell type is not *clock_network*; cells not in clock networks when cell type is *clock_network*
- UPF power management cells (cells with the *always_on*, *is_isolation*, *is_retention*, or *is_level_shifter* attribute set to *true*)
- Transparent latches
- Cells in the fanin cones of transparent latches

To allow UPF power management cells to be modified during power recovery, set the *eco_allow_sizing_with_lib_cell_attributes* variable. For example, to allow all four types of UPF power management cells to be modified:

```
set_app_var eco_allow_sizing_with_lib_cell_attributes \
  {always_on is_isolation is_retention is_level_shifter}
```

For distributed multi-scenario analysis (DMSA), set this variable in each worker process.

By default, cells in the fanin cones of transparent latches are not modified because of the difficulty of recovering the timing violations created. The default latch analysis method always sees zero slack at borrowing latches due to the single-segment nature of timing paths.

To allow downsizing and swapping cells in the fanin cones of transparent latches, enable advanced latch analysis by setting the *timing_enable_through_paths* variable to *true*. Advanced latch analysis has global visibility of timing paths through latches and can see whether a timing violation found downstream from the latch is caused by downsizing or swapping cells upstream from the latch.

Preventing Specific Cell Instances From Being Modified

Among the cells that could be downsized or swapped out, you might want to prevent specific instances from being modified. To do this, use the *set_dont_touch* command. For example,

```
set_dont_touch [get_cells {U28}] true
```

This command sets the *dont_touch* attribute to *true* for cell instance U28, which prevents the cell from being modified by ECO commands. If U28 is a hierarchical cell, the dont_touch property is propagated downward through the hierarchy and applies by default to all lower-level objects of the cell. If you apply a different dont_touch settings to specific lower-level objects, those settings override the higher-level setting.

Preventing Specific Cell Instances From Being Removed

Among the cells, you might want to prevent specific instances from being removed but allow other sizing operations on them. To do this, use the *set_size_only* command. For example,

```
set_size_only [get_cells {U28}] true
```

This command sets the *size_only* attribute to *true* for cell instance U28, which prevents the cell from being removed by ECO commands.

Preventing Specific Library Cells From Being Used

To prevent specific library cells from being used to replace existing cells, use the *set_dont_use* command. For example,

```
set_dont_use [get_lib_cells {lib2hvt/BUF3XZ}] true
```

Alternatively, you can create a user-defined lib_cell attribute named *pt_dont_use* and set it to *true*.

In distributed multi-scenario analysis (DMSA) fixing, a restriction in one scenario should also be applied to the other scenarios. Apply the same *dont_touch*, *dont_use* and *size_only* attributes to all scenarios to ensure consistent design changes across all scenarios for the affected cells and nets.

Unconstrained Cells

Unconstrained cells are cells that belong to paths without timing constraints, such as a path starting from an input port without a *set_input_delay* constraint. By default, the cell-swapping power recovery methods swap out unconstrained cells to reduce power, without considering the timing effects.

To prevent modification of unconstrained cells, set the *eco_power_exclude_unconstrained_cells* variable to *true*. In a DMSA flow, the variable must be set in the manager process. If a cell is constrained in at least one scenario, it is considered constrained in all scenarios for cell swapping purposes.

Using the cell_footprint Library Cell Attribute in ECO Fixing

If you want PrimeTime to honor the cell_footprint library cell attribute in the power recovery process, you need to import the attribute and specify the attribute name in the *eco_alternative_cell_attribute_restrictions* variable. For example, the following commands restrict the *fix_eco_power* command to swap only footprint-compatible cells.

```
# Import cell_footprint attribute to PrimeTime before linking
define_user_attribute cell_footprint -class lib_cell -import -type string

# Restrict compatible cells only to the same cell_footprint class
set eco_alternative_cell_attribute_restrictions {cell_footprint}
  ...
# The swapped-in cell must satisfy the pattern priority criterion and
# also have a matching cell_footprint string attribute
fix_eco_power -pattern_priority {HVT MVT LVT}
```

Power Recovery in Multiply Instantiated Modules (MIM)

To perform the same ECO changes across each set of multiply instantiated modules (MIMs), set the *eco_enable_mim* variable to *true*. The MIM configuration is derived from parasitic data files. If multiple instances share the same parasitics file using the *-path* option in the *read_parasitics* command, the tool recognizes them as MIM instances. For more information, see the man page for the *fix_eco_timing* command.

Unusable Cell Report

The *fix_eco_power* command, upon completion of the power recovery process, generates a report explaining why unusable cells were not used when you use the *-verbose* option.

To generate the report, the *eco_power_report_max_unusable_cells* variable must be set to a positive integer. For DMSA, this variable must be set in the manager.

Here is a report example:

```
pt_shell> fix_eco_power -verbose
  ...
Unusable Cells:
  B - Benefit from sizing cell is too small
  L - Physical constraints restrict sizing
  S - Cell has no alternate library cell with better power
  T - Sizing cell might degrade timing
  U - UPF restrictions prevent sizing
  V - Cell is invalid or has dont_touch attribute
  W - Sizing cell might degrade DRC
  X - Cell is unusable for ECO
  Z - Cell is sized

  Cell name                          Lib_cell name          Reasons
  ------------------------------------------------------------------
   U1                                      BUF1X              T W
   U3                                      BUF2X              T W
   U5                                      BUF2X                T
   U4                                      BUF1X                U
   U8                                      BUF4X                B
   U9                                      BUF8X              T W
   U7                                      BUF1X                W
  ...
```

The "Cell name" column shows the cell pins where unusable cells are located.

The "Lib_cell name" column shows library cell name.

The "Reasons" column shows the reason or reasons that the power recovery was not performed at the cell pin, using letter codes from the key shown at the beginning of the report.

Path-Based Timing Analysis During Power Recovery

The power recovery process performs timing analysis to make sure that no new timing violations are introduced, or that existing timing violations do not become worse. Path-based analysis is more accurate (less pessimistic) than conventional graph-based analysis, so it reports more positive slack and provides more opportunities for power recovery at the cost of more runtime and memory.

To invoke path-based timing analysis in the *fix_eco_power* command, use the *-pba_mode* option with the *path* or *exhaustive* setting. In that case, the tool performs more aggressive power recovery without creating new violations in the collected timing paths.

You can use the *-pba_path_selection_options* option to specify the paths collected for analysis. The path collection options are *-nworst*, *-max_paths*, and *-cover_design*. Use

-cover_design to analyze the whole design, or *-nworst* and *-max_paths* to limit the number of paths considered and save runtime. The following rules apply to these options:

- If you use the *-pba_mode path* option without the *-pba_path_selection_options* option, the tool sets the value of *-nworst* to 1 and the value of *-max_paths* to the number of graph-based analysis violations.
- If you specify *-nworst* without *-max_paths*, the tool automatically calculates and uses a value for *-max_paths* that is large enough to collect all violating endpoints.
- If you explicitly specify both the *-max_paths* and *-nworst* option values, be sure to specify numbers large enough to cover all endpoints with violations. The tool performs power recovery throughout the design, but it only checks the timing of paths included in the collection.

When you use exhaustive path-based timing analysis during power recovery, any existing violations found by path-based analysis are guaranteed not to become worse under subsequent path-based analysis. However, violations found by graph-based analysis can increase in number and become worse. This does not mean that the quality has been degraded; in only means that graph-based analysis is pessimistically reporting the results of the changes made during power recovery.

If you use the *-pba_mode exhaustive* option, the tool analyzes timing paths collected by exhaustive path-based analysis, which can significantly increase runtime and memory. In this situation, consider setting the *pba_derate_only_mode* variable to *true*, which adjusts the derating only according to path-based conditions, thereby saving runtime.

For better runtime and memory usage during path-based power recovery, consider setting the *-pba_mode* option to *path* (instead of *exhaustive*), which gives good power recovery results using the least possible path-based analysis runtime, without creating new violations detected by path-based analysis consistent with the applied *-pba_mode path* and *-pba_path_selection_options* options.

If exhaustive path-based analysis using the *report_timing* command detects any new violations after you run the *fix_eco_power* command with the *-pba_mode path* option, you can fix the violations by using the *fix_eco_timing* command with the *-pba_mode exhaustive* option.

Per-Scenario ECO Margins

In distributed multi-scenario analysis (DMSA), you can use different setup and hold margins for each scenario by specifying their values using the *-power_setup_margin* and *-power_hold_margin* options of the *set_eco_options* command in the worker script.

Note that the *-setup_margin* and *-hold_margin* options of the *fix_eco_power* command override (have higher precedence than) the *-power_setup_margin* and *-power_hold_margin* options of the *set_eco_options* command.

Physically Aware Power Recovery

Timing fixing (*fix_eco_timing*) and DRC fixing (*fix_eco_drc*) often upsize cells and insert buffers. In physically dense designs with not much available space, physically aware fixing ensures that there is room for each ECO change and tells the layout tool (such as IC Compiler II) exactly where to implement each change.

Power recovery using the *fix_eco_power* command typically replaces cells with new cells of equal or smaller size in paths with positive setup slack, or removes buffers. Therefore, even a crowded physical layout typically does not constrain power recovery. However, if physical layout information is available, the power recovery process still keeps track of physical changes so that any space freed up by downsizing or buffer removal becomes available to use later for timing fixing and DRC fixing.

If you first use the *fix_eco_timing* or *fix_eco_drc* command in physically aware mode (by setting the *-physical_mode* option to *open_site*, *occupied_site*, or *freeze_silicon*), and afterward you use the *fix_eco_power* command, the power recovery process automatically reuses and modifies the existing physical database to keep track of the recovered area. This is also true if you first use the *check_eco* command to check the physical design data.

If you have not already used another ECO fixing or checking command, but you have set up physically aware fixing with the *set_eco_options* command, the *fix_eco_power* command automatically loads the physical data and updates the physical database to keep track of the recovered area. The recovered area can be used later during timing fixing and DRC fixing. To view the chip layout in the GUI, open the GUI with the *gui_start* command and then choose Window > Layout Window.

If the *set_eco_options* command specifies one or more physical constraint files, the *fix_eco_power* command honors those constraints, such as placement blockages and library cell spacing rules.

Clock Dynamic Power Recovery

To recover clock dynamic power using PrimePower power analysis data, use the *-cell_type {clock_network}* option of the *fix_eco_power* command together with *-power_mode dynamic* and *-dynamic_scenario* options. In distributed multi-scenario analysis (DMSA) flow, the tool gets its clock dynamic power data from exactly one scenario, which you specify with the *-dynamic_scenario* option. Using these options, the tool consider cells on clock networks for downsizing when all the following conditions are met:

- The cell is a buffer or an inverter in clock networks. In a DMSA flow, the cell is in clock networks from at least one of the scenarios.
- Sizing the cell does not introduce any new hold or setup timing violation, or does not worsen existing violations, or does not cause the slack to become worse than the

margins set by the *-hold_margin* or *-setup_margin* options at all the sequential cells driven by the cell in its transitive fanout cone.

- Sizing the cell does not introduce any new DRC violations: max_transition, max_capacitance, or max_fanout.

The following options are *not* supported for clock dynamic power recovery: *-power_mode leakage*, *-method remove_buffer*, *-path_selection_options*, *-power_attribute*, *-attribute*, *-pattern_priority*, *-leakage_scenario*, and *-training_data*.

The following script performs clock dynamic power recovery in a DMSA flow with three scenarios. Scenario S1 has the worst clock dynamic power.

```
current_session {S1 S2 S3}
# update_power is only needed for power scenarios S1
current_scenario {S1}
remote_execcute {
  # Enable PrimePower and set up power analysis
  set_app_var power_enable_analysis true
  set_app_var power_clock_network_include_register_clock_pin_power false
  read_saif activity_file.saif
  # or
  # read_vcd activity_file.vcd
  ...
  update_power
}
# Report power in power scenarios before power recovery
remote_execute -verbose { report_power }
current_scenario -all

fix_eco_power -cell_type clock_network -power_mode dynamic \\
 -dynamic_scenario S1 \\

# Report power improvement in power scenarios
current_scenario {S1}
remote_execute -verbose { report_power }
```

Examples

The following command recovers area and power by downsizing both sequential and combinational cells, without introducing new timing violations (or worsening existing timing violations).

```
pt_shell> fix_eco_power
```

The following command recovers area and power by downsizing combinational cells only.

```
pt_shell> fix_eco_power -cell_type combinational
```

The following command recovers leakage power by swapping out cells with higher leakage and replacing them with cells that have lower leakage, using the library cell name prefixes to determine which cells are preferred.

```
pt_shell> fix_eco_power -pattern_priority {HVT MVT LVT}
```

The following command recovers area and power by downsizing cells and analyzing the timing effects using path-based analysis. Because path-based analysis is less pessimistic, it allows more opportunities for downsizing without introducing new timing violations.

```
pt_shell> fix_eco_power -pba_mode path
```

The following command recovers area and power by downsizing cells using path-based analysis, using the *-pba_path_selection_options* option to specify the maximum number of worst paths per endpoint considered for analysis (*-nworst* option).

```
pt_shell> fix_eco_power -pba_mode path -pba_path_selection_options
 "-nworst 10"
```

The following command recovers leakage power by replacing low-threshold-voltage cells with high-threshold-voltage cells, based on the library cell name prefixes "HVT" and "LVT", while using exhaustive path-based analysis to determine the timing effects of the changes.

```
pt_shell> fix_eco_power -pattern_priority {HVT LVT} -pba_mode exhaustive
```

The following command removes buffers to recover area or power.

```
pt_shell> fix_eco_power -methods {remove_buffer}
```

The following script performs area and power recovery in a distributed multi-scenario analysis (DMSA) flow.

```
#------------------------------------------------------------------
# Update timing & global reporting in manager process
#------------------------------------------------------------------
remote_execute {
  set timing_save_pin_arrival_and_slack true; update_timing -full
}
report_global_timing
report_constraint -all -max_capacitance -max_transition

#------------------------------------------------------------------
# Before area/power recovery:
#   Report timing, power, cell usage for each worker process
#------------------------------------------------------------------
remote_execute {
  report_global_timing
  report_constraint -all -max_capacitance -max_transition
  set power_enable_analysis true
  report_power -group {...}
  report_cell_usage
}

#------------------------------------------------------------------
# Before area/power recovery:
#   Check compatible library cells for ECO
```

Feedback

```
#-------------------------------------------------------------------
remote_execute {
  report_eco_library_cells
}

#-------------------------------------------------------------------
# Execute signoff-driven area/power recovery.
#-------------------------------------------------------------------
fix_eco_power -verbose

#-------------------------------------------------------------------
# Reporting after area/power recovery
#-------------------------------------------------------------------
report_global_timing
report_constraint -all -max_capacitance -max_transition
remote_execute {
  report_global_timing
  report_constraint -all -max_capacitance -max_transition
  set power_enable_analysis true
  report_power -group {...}
  report_cell_usage
}
exit
```

See Also

- check_eco
- get_timing_paths
- report_cell_usage
- report_eco_library_cells
- report_timing
- report_power
- set_dont_use
- set_dont_touch
- set_size_only
- set_eco_options
- update_power
- eco_allow_sizing_with_lib_cell_attributes
- eco_alternative_cell_attribute_restrictions
- eco_power_exclude_unconstrained_cells

Feedback

- pba_derate_only_mode
- power_clock_network_include_register_clock_pin_power
- timing_save_pin_arrival_and_slack
- training_data_directory

---

fix_eco_robustness

Performs design variation robustness ECO fixing.

Syntax

status *fix_eco_robustness*

```
-type bottleneck | variation_robustness | setup | hold
 | voltage_robustness | voltage_slack
[-attribute attribute_name]
[-buffer_list buffer_list]
[-cell_type combinational | sequential]
[-leakage_budget_ratio value]
[-leakage_scenario scenario_name]
[-legalize]
[-max_violators num_cells]
[-methods method_list]
[-object object]
[-pba_mode]
[-physical_mode none | open_site]
[-power_attribute attribute_name]
[-power_mode mode]
[-robustness_pattern_priority pattern_list]
[-sizing_pattern_regexp pattern_list]
[-start_end_type reg_to_reg]
[-voltage_shift value]
[-voltage_shift_ratio value]
```

Data Types

```
attribute_name   string
buffer_list      list
method_list      list
mode             string
object           string
pattern_list     list
scenario_name    string
value            float
```

-type bottleneck | variation_robustness | setup | hold | voltage_robustness | voltage_slack

Specifies the type of design violations to fix. Only one value can be specified.

-attribute *attribute_name*

Specifies an attribute name that can be used to match patterns specified by the *-robustness_pattern_priority* option. When you specify this option, the tool uses the value of the attribute instead of library cell names for pattern matching.

-buffer_list *buffer_list*

Specifies the list of library cell buffers that can be used with the *insert_buffer* method; there is no default list.

-cell_type combinational | sequential

Specifies the type of cells to be modified for timing fixing:

- *combinational* (the default) - Fixing is performed by sizing or inserting combinational logic cells in the data path.
- *sequential* - Fixing is performed by sizing sequential cells to fix violations at the sequential cell's input or output pin. Only the *size_cell* fixing method (*-methods size_cell*) is supported for this type of fixing.

-leakage_budget_ratio *value*

Specifies the maximum leakage impact allowance for a robustness ECO run. During the ECO, if the command detects that the design's leakage increment value is larger than or equal to the maximum allowance, the tool does not perform the remaining ECO steps. You must specify this option with the *-power_attribute* option. It does not work with the *-power_mode* option.

-leakage_scenario *scenario_name*

Specifies the name of the scenario from which to get the leakage power data in a DMSA flow when using the *-power_mode* option. It should be the scenario with the worst leakage power.

-legalize

Invokes the advanced legalizer to consider 3nm physical rules. This option requires an IC Compiler II reference library to be defined by the *set_eco_options* command prior to ECO fixing.

-max_violators *num_cells*

Specifies the maximum number of cells to be reported and fixed.

-method *method_list*

Specifies one or more fixing methods from the following list:

- *size_cell* - Replaces cells in the timing path with logically identical cells that have a better sensitivity or drive strength.
- *insert_buffer* - Inserts buffers in the timing path, using the library cells specified by the *-buffer_list* option.

For hold fixing, the default is *{size_cell insert_buffer}*. For other fixing types, the default is *size_cell*.

-object *object*

Specifies a *design_variation* object or a *timing_path* object.

For setup (*-type setup*), hold (*-type hold*), and bottleneck (*-type bottleneck*) fixing, you must provide a valid non-empty *design_variation* object to this option.

For voltage slack (*-type voltage_slack*) fixing, you must provide a recalculated path object that has gone through voltage slack analysis. A regular timing path collection will not trigger voltage slack ECO fixing. Other fixing types do not require the *-object* option.

-pba_mode none | path | exhaustive | ml_exhaustivess

Specifies one of the following path-based timing analysis modes:

- *none* - Disables path-based analysis and enables ordinary graph-based analysis for cell robustness and voltage robustness analysis and ECO. This is the fastest mode.
- *path* (default) – Perform on-demand exhaustive path-based analysis for cell robustness and voltage robustness analysis. This results in more accurate timing and better QoR.
- exhaustive – Perform on-demand exhaustive path-based analysis for cell robustness and voltage robustness analysis. This results in the best timing accuracy, but it is the slowest mode.
- ml_exhaustive - Perform on-demand exhaustive path-based analysis for cell robustness and voltage robustness analysis. The uses machine-learning exhaustive PBA to trade runtime versus accuracy during the ECO flow. Accuracy and runtime will match the *-pba_mode exhaustive* option.

For fixing types *setup*, *hold*, *bottleneck*, and *voltage_slack*, the *-pba_mode* option does not affect the PrimeShield analysis results, but it does affect how the ECO fixing engine makes design changes while considering graph-based or path-based timing.

-physical_mode none | open_site

Specifies the usage of physical data to size cells or place buffers:

- *none* (the default) - Robustness fixing does not use physical data.
- *open_site* - Robustness fixing sizes a cell only if there is enough room available around the cell and inserts a buffer only if there is an empty site with enough room to accept the new cell without moving nearby cells. This mode retains the placement of existing cells.

-power_attribute *power_attribute_name*

Specifies a user-defined attribute for library cells that defines the leakage power of the library cell at the worst operating condition corner for leakage. When you use this option, timing fixing attempts to minimize the increase in leakage by giving higher priority to library cells with less leakage. By default, without this option, timing fixing considers area instead of leakage.

For fixing types setup (*-type setup*), hold (*-type hold*), and voltage_slack (*-type voltage_slack*), you can specify the *-power_attribute* option alone to achieve power-aware robustness ECO fixing. With the *-leakage_budget_ratio* option, you can achieve even more precise control over the power impact.

For fixing types bottleneck (*-type bottleneck*), variation_robustness (*-type variation_robustness*) and voltage_robustness (*-voltage_robustness*), you must use both *-power_attribute* and *-leakage_budget_ratio* to achieve power-aware ECO fixing.

-power_mode *mode*

Specifies the use of PrimePower power analysis data and specifies the types of power to consider during timing fixing. The only supported value is *leakage*.

Power analysis data must be available from a previous *update_power* or *report_power* command. You cannot use the *-power_mode* option with the *-power_attribute* option. This option applies only to the *setup*, *hold*, and *voltage_slack* fixing types.

When you use the *-power_mode* option for *setup* fixing, the tool tries to minimize the increase in the specified type of power by giving higher priority to library cells that use less power.

Similarly, for *hold* fixing, the tool prioritizes the library cells with lower power during cell sizing and buffer insertion.

By default, without the *-power_mode* option, robustness fixing considers area instead of power.

-robustness_pattern_priority *pattern_list*

Specifies a list of library cell patterns, from highest to lowest priority. The *-robustness_pattern_priority* option is required for *bottleneck*, *variation_robustness*, and *voltage_robustness* fixing, but it is optional for *setup* and *voltage_slack* fixing. The ECO strictly follows the provided pattern to perform pattern-based cell swapping.

-sizing_pattern_regexp *pattern_list*

Specifies a regular expression pattern that can be used to match library cells with different cell drive strengths.

This option only applies to the *bottleneck*, *variation_robustness*, and *voltage_robustness* fixing types.

-start_end_type *reg_to_reg*

Restricts ECO fixing to register-to-register paths. This option is supported only for the *voltage_robustness* and *variation_robustness* ECO fixing types.

-voltage_shift *value*

Specifies the voltage drop used for the sensitivity calculation, as an absolute value in volts. The default is a 5% relative drop from each cell's nominal VDD value. You should use the same value used for the *report_cell_robustness -type voltage* option.

-voltage_shift_ratio *value*

Specifies the voltage drop used for the sensitivity calculation, as a relative ratio (such as 0.95) of each cell's nominal VDD value. The default is a 5% relative drop. You should use the same value used for the *report_cell_robustness -type voltage -voltage_shift_ratio* option.

Description

The *fix_eco_robustness* command improves design variation by performing VT-swap, sizing, and buffer insertion operations without introducing or worsening timing and DRC violations. Because timing-critical cells and endpoints also affect the robustness analysis, you should run *fix_eco_robustness* after timing, DRC, and power fixing to avoid over-fixing. However, you can perform *variation_robustness* and *voltage_robustness* fixing in the early stage to minimize the number of weak cells in the design to improve signoff quality. By default, ECO fixing uses path-based timing analysis for improved accuracy.

You can report the existing violations using the *report_variation_bottleneck*, *report_cell_robustness*, *report_design_variation*, and *report_timing -voltage_slack_lesser_than* commands. The *fix_eco_robustness* command attempts to fix the violations while minimizing the impact on the area or power.

In general, VT swapping is more effective in improving a design's variation robustness while retaining the design's area characteristics. However, swapping to a lower VT usually leads to an increase in leakage power. To address this, the *fix_eco_robustness* command has a power-aware flow to minimize the ECO impact on the power.

After robustness ECO fixing, you can write out the design changes as a script by using the *write_changes* command. You can use this script to implement the same changes in another PrimeTime session, or in another tool such as the IC Compiler II tool. The list of design changes created in this flow is called an engineering change order (ECO).

Setup Fixing

The *fix_eco_robustness* command attempts to improve the timing slack of the setup variation critical endpoints in order to achieve a 99.865% (3 sigma) joint success rate (JSR) in setup timing. You must run the *get_design_variation* and *report_design_variation* commands before performing setup robustness ECO fixing. The *fix_eco_robustness* command takes the object argument by variable reference. For example,

```
set dvar [get_design_variation -sample_size 100000 $max_path]
report_design_variation -target_success_rate 0.99865 $dvar
fix_eco_robustness -type setup -object dvar
```

The default fixing method for setup fixing is *size_cell*. ECO fixing replaces cells with lower variation cells or higher drive strength cells to improve the path's variability. You can specify VT patterns by using the *-robustness_pattern_priority* option. This ensures the ECO only performs pattern-based VT swap.

Hold Fixing

The *fix_eco_robustness* command attempts to improves the timing slack of the hold variation critical endpoints in order to achieve a 99.865% (3 sigma) joint success rate (JSR) in hold timing. You must run the *get_design_variation* and *report_design_variation* commands before hold robustness ECO fixing. The *fix_eco_robustness* command takes the object argument by variable reference. For example,

```
set dvar [get_design_variation -sample_size 100000 $min_path]
report_design_variation -target_success_rate 0.99865 $dvar
fix_eco_robustness -type hold -object dvar \\
  -buffer_list {BUFFD1 BUFFD2 BUFFD6 BUFFD8}
```

The default fixing methods for hold fixing are both *size_cell* and *insert_buffer*. Both methods aim to improve the hold timing JSR by improving the worst timing slack of the hold critical endpoints. The *-robustness_pattern_priority* option does not apply to hold fixing.

Bottleneck Fixing

The *fix_eco_robustness* command tries to reduce the design's High Sigma Failure Rate (HSFR) by VT swap and cell sizing. Variation bottleneck analysis applies only to setup

timing; it does not apply to hold timing. You must run the *get_design_variation* and *report_variation_bottleneck* commands before performing robustness bottleneck ECO fixing. The command takes the object argument by variable reference. For example,

```
set dvar [get_design_variation -sample_size 100000 $max_path]
report_variation_bottleneck -max_violators 50000 -pba
fix_eco_robustness -type bottleneck -object dvar \\
  -robustness_pattern_priority {LVT SVT HVT}
```

The default fixing method is *size_cell*. You must specify one or both of the *-robustness_pattern_priority* and *-sizing_pattern_regexp* options. If both options are provided, ECO fixing prioritizes VT swap over cell sizing.

Cell Robustness Fixing

Cell robustness analysis uses the same High Sigma Failure Rate calculation algorithm as the variation bottleneck analysis, except that it does not require the *get_timing_path* and *get_design_variation* commands. Therefore, you do not need to provide an object argument to the *fix_eco_robustness* command. For example,

```
fix_eco_robustness -type variation_robustness \\
  -robustness_pattern_priority {LVT SVT HVT}
```

The default fixing method is *size_cell*. You must specify one or both of the *-robustness_pattern_priority* and *-sizing_pattern_regexp* options. If both options are provided, ECO fixing prioritizes VT swap over cell sizing.

Voltage Robustness Fixing

Voltage robustness ECO targets the weak cells that are sensitive to voltage drop in the design that could lower a path's voltage slack. After running voltage robustness ECO fixing, an average voltage slack improvement is expected; that is, the average post-ECO voltage slack is greater than the average pre-ECO voltage slack.

```
fix_eco_robustness -type variation_robustness \\
  -robustness_pattern_priority {LVT SVT HVT}
```

The default fixing method is *size_cell*. You must specify one or both of the *-robustness_pattern_priority* and *-sizing_pattern_regexp* options. If both options are provided, ECO fixing prioritizes VT swap over cell sizing.

Voltage Slack Fixing

Voltage slack ECO prioritizes fixing paths with smaller voltage slack to improve the design's resilience to IR drops. You must first define a voltage target by setting the *eco_voltage_slack_target* variable before using the *get_timing_paths* command. You must also provide the path object to the command by reference using the *-object* option. ECO

fixing attempts to improve the voltage slack of the given path to be greater than or equal to the specified target voltage slack value. For example,

```
set eco_voltage_slack_target 0.02
set path [get_timing_path -vdd_slack_lesser_than 0.02 \\
  -pba_mode exhaustive -path full_clock_expanded]
fix_eco_robustness -type voltage_slack -object path
```

The default fixing method is *size_cell*. ECO fixing replaces cells on the critical path with lower variation or bigger drive strength cells to improve the voltage slack. You can specify a VT pattern by using the *-robustness_pattern_priority* option. This ensures that ECO fixing performs only pattern-based VT swaps.

Pattern Priority and Sizing Pattern Regular Expression

The *bottleneck*, *variation_robustness*, and *voltage_robustness* fixing types require one or both of the *-robustness_pattern_priority* and *sizing_pattern_regexp* options. The *fix_eco_robustness* command uses the provided pattern for pattern-based sizing. Setup and voltage slack can only accept the *-robustness_pattern_priority* option.

For the *-robustness_pattern_priority* option, specify the VT patterns in the order of lowest variation VT type to highest variation VT type.

Assume the following three types of library cells are available in the design:

```
*/BUFF_D1_L10_UL
*/BUFF_D1_L10_L
*/BUFF_D1_L10_S
```

The pattern "UL" represents the lowest variation whereas "S" represents the highest variation. Set the pattern as follow:

```
fix_eco_robustness -type variation_robustness \\
  -robustness_pattern_priority {_L10_UL _L10_L _L10_S}
```

The tool swaps the critical cell to an "L10_L" cell if it is an "L10_S" cell. It swaps to an "L10_UL" cell if it is an "L10_L" cell. The tool only swaps one type up in every ECO iteration, which means it never swaps from an L10_S cell to an L10_UL cell in one iteration.

Avoid specifying a non-unique pattern list such as {UL L S}. This is because the command uses a simple wildcard matching technique to match the patterns to the lib cells names. You can validate the patterns by using the command *report_eco_library_cells -robustness_pattern_priority*, which reports alternative library cells with the given pattern.

For the *-sizing_pattern_regexp* option, specify the sizing pattern in the regular expression. Assume the following three types of library cells are available in the design:

```
*/BUFF_D1_L10_UL
*/BUFF_D2_L10_UL
*/BUFF_D3_L10_UL
```

"D1" represents the lowest drive strength and "D3" represents the highest drive strength. Specify the command as follows:

```
fix_eco_robustness -type variation_robustness \\
  -sizing_pattern_regexp {_D[0-9]+_}
```

You can specify both the *-robustness_pattern_priority* and *-sizing_pattern_regexp* options concurrently. In this case, ECO fixing uses the VT swap method first, using the priority pattern, until it reaches the lowest VT pattern. Then, it performs cell sizing using the sizing pattern.

```
fix_eco_robustness -type variation_robustness \\
  -sizing_pattern_regexp {_D[0-9]+_} \\
  -robustness_pattern_priority {_L10_UL _L10_L _L10_S}
```

If needed, you can specify multiple pattern sets separated by ";":

```
fix_eco_robustness -type variation_robustness \\
  -robustness_pattern_priority {_L10_UL _L10_L _L10_S;_LL _SS}
fix_eco_robustness -type variation_robustness \\
  -sizing_pattern_regexp {_D[0-9]+_;D_[0-9]+}
```

Information messages indicate the successful configuration of the VT and sizing patterns.

```
...
Information: Found 591 sets of vt patterns.
Information: Found 449 sets of sizing patterns.
 ...
```

Power-Aware Robustness ECO

There are two power-aware options for *fix_eco_robustness*: *-power_attribute* and *-power_mode*.

The *fix_eco_robustness* command supports the *-power_attribute* for all ECO fixing types. ECO fixing optimizes power during robustness fixing based on the defined power attribute value. For example, create a user-defined attribute "leak_attr" for lib_cell object:

```
# Create a user-defined attribute "leak_attr" for lib_cell object
define_user_attribute leak_attr -type float -class lib_cell

# Assign values that reflect leakage of each lib_cell at worst corner
set_user_attr -class lib_cell [get_lib_cell LIB_H/INV1X_HVT] leak_attr
 1.0
set_user_attr -class lib_cell [get_lib_cell LIB_H/INV2X_HVT] leak_attr
 2.0
set_user_attr -class lib_cell [get_lib_cell LIB_L/INV1X_LVT] leak_attr
 10.0
set_user_attr -class lib_cell [get_lib_cell LIB_L/INV2X_LVT] leak_attr
 15.0
 ...
```

The *-power_attribute* option can be specified with the *-leakage_budget_ratio* option, which controls the maximum power impact a design can have after ECO fixing. For *variation_robustness*, *voltage_robustness*, and *bottleneck* fixing types, you must use the *-leakage_budget_ratio* option together with the *-power_attribute* option. For the *setup*, *hold*, and *voltage_slack* fixing types, the *-leakage_budget_ratio* option is optional because these fixing types use the same power optimization technique as in the *fix_eco_timing* command.

```
# Run power-aware voltage slack ECO
fix_eco_robustness -type voltage_slack \\
  -power_attribute leak_attr

# Run power-aware voltage slack ECO with maximum 2% power impact
fix_eco_robustness -type voltage_slack \\
  -power_attribute leak_attr -leakage_budget_ratio 0.02

# Run cell robustness ECO with maximum 1% power impact
fix_eco_robustness -type variation_robustness \\
  -power_attribute leak_attr -leakage_budget_ratio 0.01
```

Only the *setup*, *hold*, and *voltage_slack* fixing methods support the *-power_mode* leakage option. With the *-power_mode* leakage option, the tool fixes the violations considering leakage power without requiring user-defined power attributes. This mode requires the power data to be ready by running *update_power* or *report_power* before ECO fixing.

```
set power_enable analysis true
report_power
fix_eco_robustness -power_mode leakage -leakage_scenario tt_lkg
```

Physically-Aware Robustness ECO

Set the *-physical_mode* option to *open_site* to enable the command to load physical data into memory and use that data while fixing robustness violations. Before you run the *fix_eco_robustness* command, use the *set_eco_options* command to specify the paths to the physical data, either IC Compiler II database files or LEF/DEF library and design data exchange files.

To consider 3nm physical rules, you must enable the advanced legalization mode by using the *-legalize* option. The advanced legalizer requires IC Compiler II reference library in the seesion, and the tool errors out if the physical data is not available.

All physically-aware robustness ECOs are in-place changes, which means that the tool searches free space and upsizes the instance in the same location without moving surrounding cells.

Fixing Summary

After completing *variation_robustness*, *bottleneck*, or *voltage robustness* ECO, the command prints an ECO summary. In the summary, the pre-ECO values such as the "Total cell found" and "Total slack found" are the total violators found and total slack found in all

ECO iterations. Other values, like "Total percentage fixed" and "Total cell remaining", are estimated. These estimated numbers are generally very close compared to the golden numbers found in the post-ECO analysis. The prediction algorithm is designed to estimate the ECO trend without an extra time cost. It is highly recommended that you run a post-ECO analysis to get the golden values.

```
Fixing Summary:
-------------------------------------------------------
Total robustness cell found                        66
Total robustness cell fixed                        63
Total robustness cell remaining                    3
Total percentage of cells fixed                    95.45%

Total robustness_slack found                       -4.56
Total robustness_slack remaining                   -0.44
Total percentage of robustness_slack reduced       90.35%
```

Unfixable reasons

After running ECO fixing for the *variation_robustness*, *bottleneck*, and *voltage_robustness* fixing types, the command generates a summary to show the unfixable reasons for the remaining cells. Below are the unfixable messages and the corresponding definitions:

```
01) Not covered in patterns:        The violator is not covered in either
 VT or sizing pattern
02) Dont touch:                     The violator is a dont_touch cell
03) Clock cell:                     The violator is a clock component
06) Lowest pattern already:         Current library cell is the lowest
 pattern already
07) No available candidates:        All swappable cells have a dont_use
 attribute
08) Hold critical:                  The violator has hold timing
 violations. ECO can further degrade hold timing.
09) Physical rule:                  ECO violates physical rules
10) Hold timing degrade:            ECO can cause hold timing degradation
11) Setup timing degrade:           ECO can cause setup timing
 degradation
12) Cells attributes restriction:   ECO doesn't meet attribute
 restriction
14) Not enough space:               Not enough space for sizing
16) Sequential cell:                The violator is a sequential cell
17) Combinational cell:             The violator is a combination cell
19) Advanced legalizer:             ECO violates N3 physical rules
20) Leakage budget limit:           ECO exceeds maximum leakage impact
 allowance
21) Cannot be further improved:     Cannot further improve robustness
 slack
```

Feedback

Distributed Multi-Scenario Analysis (DMSA)

Run the *fix_eco_robustness* command in the DMSA manager only. If you have multiple scenarios, the command requires at least one scenario that has robustness violations. Other scenarios will be considers for timing impact only during ECO fixing.

Examples

The following example fixes design variation bottleneck cells using the VT swap. The "LVT" type cells have the lowest variation. The "HVT" type cells have the highest variation. The VT pattern can be expressed as {LVT SVT HVT} in the order of lowest sensitivity to highest sensitivity.

```
prompt> set path [get_timing_path -pocv_pruning -pba_mode exhaustive \\
          -path_type full_clock_expanded \\
          -nworst 10 -max_paths 10000 \\
          -slack_lesser_than 1.0]
prompt> set dvar [get_design_variation -sample_size 100000 $max_path]
prompt> report_variation_bottleneck -max_violators 10000
prompt> fix_eco_robustness -type bottleneck -object dvar \\
          -robustness_pattern_priority {LVT SVT HVT}
```

The following example fixes voltage robustness cells with 10% voltage drop using a sizing pattern. The sizing pattern can be expressed as "_D[0-9]+_" in regular expression form.

```
prompt> report_cell_robustness -type voltage \\
          -max_violators 10000 -pba \\
          -voltage_shift_ratio 0.1
prompt> fix_eco_robustness -type voltage_robustness \\
          -sizing_pattern_regexp {_D[0-9]+_} \\
          -voltage_shift_ratio 0.1
```

The following example fixes cell robustness cells found in register-to-register paths using the VT swap. The VT pattern can be expressed as {LVT SVT HVT} in the order of lowest sensitivity to highest sensitivity.

```
prompt> report_cell_robustness -type variation_robustness \\
          -max_violators 10000 -pba -start_end_type reg_to_reg
prompt> fix_eco_robustness -type variation_robustness \\
                    -robustness_pattern_priority {LVT SVT HVT} \\
                    -start_end_type reg_to_reg
```

The following example shows how to fix setup design variation violations with the power-aware flow using user-defined power attribute values

```
prompt> report_cell_usage -power_attr pwr_attr
prompt> set path [get_timing_path -pocv_pruning -pba_mode exhaustive \\
          -path_type full_clock_expanded \\
          -nworst 10 -max_paths 10000 \\
          -slack_lesser_than 1.0]]
prompt> set dvar [get_design_variation -sample_size 100000 $max_path]
```

Feedback

```
prompt> fix_eco_robustness -type setup -object dvar \\
          -power_attribute pwr_attr \\
          -robustness_pattern_priority {LVT SVT HVT}
```

The following example shows how to run physical-aware voltage slack ECO with the target slack equals 0.02V.

```
prompt> set_eco_options \\
          -physical_tech_lib_path $LEF_tech_file_list \\
          -physical_lib_path $LEF_lib_file_list \\
          -physical_design_path $DEF_design_file_list

## This variable must be set before get_timing_path, otherwise the ECO
 will error out
prompt> set eco_voltage_slack_target 0.02
prompt> set path [get_timing_path -vdd_slack_lesser_than 0.02 \\
           -pba_mode exhaustive \\
           -path_type full_clock_expanded \\
           -nworst 10 -max_paths 10000 \\
           -slack_lesser_than 1.0]
prompt> fix_eco_robustness -type voltage_slack \\
           -object path -physical_mode open_site
```

The following example shows how to run voltage slack ECO in DMSA mode.

```
prompt> set eco_voltage_slack_target 0.02
prompt> remote_execute -v { \\
          set eco_voltage_slack_target 0.02
          set path [get_timing_path -vdd_slack_lesser_than 0.02 \\
            -pba_mode exhaustive \\
            -path_type full_clock_expanded \\
            -nworst 10 \\
            -max_paths 10000 \\
            -slack_lesser_than 1.0]
           }
prompt> fix_eco_robustness -type voltage_slack \\
          -object path -physical_mode open_site
```

See Also

- get_timing_paths
- get_design_variation
- report_variation_bottleneck
- report_cell_robustness
- report_voltage_robustness

Feedback

- ps_enable_analysis
- eco_voltage_slack_target

---

fix_eco_timing

Fixes or improves timing violations through engineering change orders (ECOs).

Syntax

```
status fix_eco_timing
   -type setup | hold
   [-methods method_list]
   [-slack_lesser_than slack_limit]
   [-slack_greater_than slack_limit]
   [-group group_name]
   [-pba_mode none | path | exhaustive | ml_exhaustive]
   [-from from_list]
   [-to to_list]
   [-setup_margin margin]
   [-hold_margin margin]
   [-buffer_list list]
   [-verbose]
   [-current_library]
   [-ignore_drc]
   [-cell_type combinational | sequential | clock_network]
   [-physical_mode none | open_site | occupied_site | freeze_silicon]
   [-path_selection_options option_string]
   [-timeout seconds]
   [-power_attribute power_attribute_name]
   [-clock_fixes_per_change violation_limit]
   [-clock_max_level_from_reg level_limit]
   [-estimate_unfixable_reasons]
   [-power_mode total | dynamic | leakage]
   [-leakage_scenario scenario_name]
   [-dynamic_scenario scenario_name]
   [-load_cell_list list]
   [-target_violation_type violation_type]
   [-wns_limit wns_value]
   [-unfixable_reasons_format text | csv]
   [-unfixable_reasons_prefix string]
```

Data Types

```
method_list          list
slack_limit          float
group_name           string
from_list            list
to_list              list
margin               float
list                 list
option_string        string
```

```
seconds                integer
power_attribute_name   string
violation_limit        integer
level_limit            integer
scenario_name          string
violation_type         string
wns_value              float
string                 string
```

Arguments

`-type setup | hold`

Specifies fixing of either *setup* (maximum delay) or *hold* (minimum delay) timing violations.

`-methods method_list`

Specifies one or more fixing methods from the following list:

- *size_cell* - Replaces cells in the timing path with logically identical cells that have a different drive strength.
- *size_cell_side_load* - Same as *size_cell*, but also replaces cells in the fanout of drivers in the path with logically identical cells to reduce parasitic load capacitance along the path (used only for *-type setup* fixing).
- *insert_buffer* - Inserts buffers in the timing path, using the library cells specified by the *-buffer_list* option. By default, the *insert_buffer* method considers buffer insertion at driver pins, load pins, and on-route.
- *insert_buffer_at_load_pins* - Inserts buffers only at load pins, not at driver pins or on-route.
- *insert_buffer_at_driver_pins* - Inserts buffers only at driver pins, not at load pins or on-route.
- *insert_inverter_pair* - Inserts pairs of inverter cells instead of buffer cells; cannot be used together with the *insert_buffer* method
- *bypass_buffer* - Bypasses buffers or inverter pairs in a clock network; can be used only when the *clock_network* cell type is specified.
- *remove_buffer* - Removes buffers to improve setup timing without introducing or worsening hold timing and DRC violations.

For setup fixing (*-type setup*), you can specify either *size_cell* or *size_cell_side_load*; the default is *{size_cell_side_load}*. You can use the *insert_buffer* method alone or together with *size_cell* or *size_cell_side_load* if the *-physical_mode* option is set to either *open_site* or *occupied_site*; for the *freeze_silicon* setting, *insert_buffer* is the only method allowed. To insert buffers

only at driver pins, use the *insert_buffer_at_driver_pins* method without the *insert_buffer* method.

For hold fixing (*-type hold*), you can specify one or more of the methods *size_cell*, *insert_buffer*, *insert_buffer_at_load_pins*, and *insert_inverter_pair*; the default is *{size_cell insert_buffer}*. To insert buffers only at load pins, use the *insert_buffer_at_load_pins* option without the *insert_buffer* option.

The *insert_inverter_pair* method inserts inverter pairs both at driver and load pins, choosing from the inverter library cells specified by the *-buffer_list* option. This method works well together with *size_cell* (*-methods {size_cell insert_inverter_pair}*).

The *remove_buffer* method has the following restrictions:

- It cannot be used with any of the following options: *-cell_type*, *-power_attribute*, *-current_library*, *-load_cell_list*, *-timeout*, *-unfixable_reasons_format*, *-unfixable_reasons_prefix*, and *-power_mode*.
- In setup fixing, it cannot be combined with other methods; it must be used by itself.
- It cannot be used in hold fixing.

`-slack_lesser_than` *`slack_limit`*

Specifies fixing of only the paths with a slack worse than the specified slack limit. The default is zero, which attempts to fix all timing violations.

`-slack_greater_than` *`slack_limit`*

Specifies fixing of only the paths with a slack better than the specified slack limit. By default, the command tries to fix all timing violations. You can use this option to avoid consideration of paths with very negative slacks caused by known design problems or constraints, while still considering less extreme violations.

You can combine the *-slack_lesser_than* and *-slack_greater_than* options to fix paths that have a slack within a specified range.

`-group` *`group_name`*

Restricts fixing to paths that belong to the specified path groups. By default, paths are divided into groups according to the clock that captures data at the path endpoint; a path group is created by each *create_clock* command. You can also create path groups explicitly by using the *group_path* command. You can specify multiple path groups by using a list of group names (wildcards allowed) or a collection created by the *get_path_groups* command.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Specifies one of the following timing analysis modes:

- *none* (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- *path* - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths.
- *exhaustive* - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode.
- *ml_exhaustive* - Performs machine learning based exhaustive path-based analysis. This mechanism will deploy machine learning techniques to trade runtime vs accuracy during the design flow. Accuracy and runtime will match the *-pba_mode exhaustive* option as the design approaches signoff.

This option works like the *-pba_mode* option in the *report_timing* and *get_timing_paths* commands. The fixing process uses the specified analysis mode to find the timing violations that need to be fixed.

`-from` *`from_list`*

Specifies a list of "from" pins or ports. Only the timing paths that start from the specified objects are considered for fixing.

`-to` *`to_list`*

Specifies a list of "to" pins or ports. Only the timing paths that end at the specified objects are considered for fixing.

The *-from* and *-to* options are like the options of the same name in the *report_timing* and *get_timing_paths* commands, except that you can only specify pins or ports, not nets or cells, as the objects. The default behavior is to consider all pins and ports, and to select the path with the worst slack within each path group.

`-setup_margin` *`margin`*

Specifies an additional margin for setup fixing, in library time units. The default margin is zero. This option affects the target amount of timing slack to achieve during setup fixing, or the minimum amount of setup margin to maintain during hold fixing. Use this option together with the *-slack_lesser_than* option, specifying a positive slack value, to target fixing of paths with positive slack and obtain the desired margin.

During setup fixing, a positive margin specifies overfixing, which attempts to achieve the specified positive slack value. A negative margin specifies

underfixing, which attempts to improve the timing and achieve the specified negative slack value. For paths not already selected for setup fixing (by default, paths with zero or better slack), there is no attempt to change the path to meet the margin requirement; only paths already targeted for setup fixing are modified.

During hold fixing, the tool attempts to preserve the specified setup margin while making changes to fix hold violations.

`-hold_margin` *`margin`*

Specifies an additional margin for hold fixing, in library time units. The default margin is zero. This option affects the target amount of timing slack to achieve during hold fixing, or the minimum amount of hold margin to maintain during setup fixing. Use this option together with the *-slack_lesser_than* option, specifying a positive slack value, to target fixing of paths with positive slack and obtain the desired margin.

During hold fixing, a positive hold margin specifies overfixing, which attempts to achieve the specified positive slack value. A negative margin specifies underfixing, which attempts to improve the timing and achieve the specified negative slack value. For paths not already selected for hold fixing (by default, paths with zero or better slack), there is no attempt to change the path to meet the margin requirement; only paths already targeted for hold fixing are modified.

During setup fixing, the tool attempts to preserve the specified hold margin while making changes to fix setup violations.

In the absence of explicit setup and hold values provided by the user, the tool tries to preserve DRC and timing by default, however, the results might not match explicit settings of *-setup_margin 0.0* and *-hold_margin 0.0*.

`-buffer_list` *`buffer_list`*

Specifies the list of library cell buffers that can be used with the *insert_buffer* method or inverters that can be used with the *insert_inverter_pair* method. To use the *insert_buffer* or *insert_inverter_pair* method, you must also use the *-buffer_list* option to specify the allowed library cells; there is no default list.

Specify the list of buffers using simple library cell base names, without the library name. The command gets the buffer cells from the libraries listed in the *link_path* variable. It searches the libraries in order and uses the first library cell found with a matching name.

The command can use a listed library cell for buffer insertion even if its *dont_use* attribute is set to *true*. The *dont_use* attribute applies only to cell sizing.

In hold fixing, if you use *-methods {size_cell insert_buffer}*, the command can insert a buffer from the list and then size it to a different buffer, even if the sized library cell is not explicitly included in the *-buffer_list* list.

`-verbose`

Shows additional information during the fixing process. In distributed multi-scenario analysis, violation information from each scenario is displayed.

The verbose mode also generates an "Unfixable Violations" report upon completion of fixing if the *eco_report_unfixed_reason_max_endpoints* variable is set to a positive integer. The report shows a list of pins where fixing was considered, the reason that the fix was not performed at the pin, and the fixing priority for the pin. For details, see "Unfixable Violation Report" in the DESCRIPTION section of this man page. To generate the report without actually performing any fixing, use the *-estimate_unfixable_reasons* option.

`-current_library`

Specifies the usage of library cells from the same library when performing cell sizing. With this option, the replacement cell must come from the same library as the original cell. This option can improve runtime because fewer cells are considered as potential replacements. However, it might degrade quality of results because the variety of available alternative library cells could be reduced.

`-ignore_drc`

Causes timing fixing to ignore DRC violations. By default, timing fixing is not performed when there are any existing DRC violations on the path (max_transition, max_capacitance, and max_fanout violations). With this option, the command proceeds with timing fixing without considering DRC violations, possibly worsening those violations.

`-cell_type combinational | sequential | clock_network`

Specifies the type of cells to be modified for timing fixing:

- *combinational* (the default) - Fixing is performed by sizing or inserting combinational logic cells in the data path.
- *sequential* - Fixing is performed by sizing sequential cells to fix violations at the sequential cell's input or output pin. Only the *size_cell* fixing method (*-methods size_cell*) is supported for this type of fixing.
- *clock_network* - Fixing is performed by sizing, inserting combinational cells, or bypassing combinational cells in the clock network, thereby changing clock arrival times.

You can specify just one of these options, or both *combinational* and *sequential* together to fix both cell types simultaneously.

The *sequential* and *clock_network* type modifications can adversely affect clock tree timing, so you should try *combinational* type fixing first and only apply the other types afterward if needed.

`-physical_mode none | open_site | occupied_site | freeze_silicon`

Specifies the usage of physical data to size cells or place buffers:

- *none* (the default) - Timing fixing does not use physical data. For setup fixing using buffer insertion (*-type setup -methods {insert_buffer ...}*), you must use physical data by choosing one of the other physical mode options.
- *open_site* - Timing fixing sizes a cell only if there is enough room available around the cell and inserts a buffer only if there is an empty site with enough room to accept the new cell without moving nearby cells. This mode retains the placement of existing cells.
- *occupied_site* - Timing fixing can size a cell and insert a buffer that overlaps existing neighbor cells, as long as the cell density (area utilization) is low enough that the nearby cells can be moved to make the required space. After the change is made, you need to use a physical implementation tool such as IC Compiler II to move the existing cells and create room for the sized or inserted cell.
- *freeze_silicon* - Timing fixing uses spare cells defined in the physical data files and identified by the *-programmable_spare_cell_names* option of the *set_eco_options* command; it does not insert new cells or size existing cells. This mode works with the IC Compiler II physical implementation tool. For details, see "Physically Aware Freeze Silicon Flow" in the DESCRIPTION section.

`-path_selection_options` *`option_string`*

Specifies the path selection options for the *get_timing_paths* command, which the *fix_eco_timing* command uses to collect timing paths for fixing. Because fixing is performed only on these paths, to check the quality of results after fixing, use the *get_timing_paths* or *report_timing* command with the same options.

The *-path_selection_options* option cannot be used together with the *-from*, *-to*, *-group*, *-pba_mode*, *-slack_lesser_than*, or *-slack_greater_than* options.

The *-path_selection_options* option

- Supports only the options of the *get_timing_paths* command that are relevant to ECO timing fixing.
- Must be enclosed with curly braces { } when it contains any collection.
- Must explicitly use the *-max_paths* and *-nworst* options with values that are suitable for timing fixing unless either the *-start_end_pair* or *-cover_design* option is used.

`-timeout` *`seconds`*

Specifies a timeout limit, in seconds, that sets a maximum total runtime for the command. At the end of each fixing iteration, the tool checks the total elapsed wall clock time. If the time limit is reached, the command stops running and ends with the current iteration. If you do not use this option, the command does not directly consider the total elapsed time. Instead, it considers quality of results, fix rate, and expected benefits of running more iterations.

`-power_attribute` *`power_attribute_name`*

Specifies a user-defined attribute for library cells that defines the leakage power of the library cell at the worst operating condition corner for leakage. When you use this option, timing fixing attempts to minimize the increase in leakage by giving higher priority to library cells with less leakage. By default, without this option, timing fixing considers area instead of leakage.

`-clock_fixes_per_change` *`violation_limit`*

Specifies the minimum number of violations to be fixed per cell sizing, buffer insertion, or inverter pair insertion in the clock network. This setting has an effect only if the *-cell_type* option is set to *clock_network*. The default is 1. Set this option to a value larger than 1 to reduce the number of changes made to the clock network.

For example, if you set this option to 4, the command makes a change in the clock network only if the change fixes at least 4 violating endpoints at leaf sequential cells in the transitive fanout of the change. Setting a larger value reduces the number of changes and also forces them to occur at higher levels of the clock network, closer to the clock source and farther from the sequential cells.

`-clock_max_level_from_reg` *`level_limit`*

Specifies the highest level of a clock network tree in which timing fixes can occur. The higher the level is, the closer the fix is performed to the clock source. This option has an effect only if the *-cell_type* option is set to *clock_network*. The default is 0, which disables the limit and allows changes anywhere in the clock network.

Setting this option to 1 allows changes to be made only at the immediate driver cell to a clock pin of a sequential cell. In this case, the command can size the immediate driver or insert a buffer between this driver and the clock pin of the sequential cell. Setting this option to 2 allows the command to make these changes at both the immediate driver and one buffer level above the immediate driver; and so on for larger settings.

`-estimate_unfixable_reasons`

Performs a fixing analysis (without actually making any changes) and generates a report on violations that are unlikely to be fixed and the reasons that they are unfixable. This option is similar to the *-verbose* option, except that no actual fixing is performed; only the report is generated. To use this option, the *eco_report_unfixed_reason_max_endpoints* variable must be set to a positive integer (the default is 0). For details, see "Unfixable Violation Report" in the DESCRIPTION section.

`-power_mode total | dynamic | leakage`

Specifies usage of PrimePower power analysis data, and specifies the types of power considered during timing fixing: *dynamic* power only, *leakage* power only, or *total* power (both dynamic and leakage). Power analysis data must be available from previous usage of the *update_power* or *report_power* command. You cannot use the *-power_mode* option with the *-power_attribute* or *-cell_type clock_network* option.

When you use the *-power_mode* option for setup fixing, the tool tries to minimize the increase in the specified type of power by giving higher priority to library cells that use less power. This includes side-load sizing by the *-methods size_cell_side_load* option.

Similarly, for hold fixing, the tool prioritizes the library cells with lower power during cell sizing and buffer insertion.

By default, without the *-power_mode* option, timing fixing considers area instead of power.

`-leakage_scenario` *`scenario_name`*

Specifies the name of the scenario from which to get the leakage power data in a DMSA flow when using the *-power_mode* option. It should be the scenario with the worst leakage power.

`-dynamic_scenario` *`scenario_name`*

Specifies the name of the scenario from which to get the dynamic power data in a DMSA flow when using the *-power_mode* option. It should be the scenario with the worst dynamic power.

The *-leakage_scenario* and *-dynamic_scenario* options can be set to the same scenario or two different scenarios.

`-load_cell_list` *`list`*

Specifies a list of library load capacitance cells, buffer cells, or inverter cells that can be used along the data path for hold fixing with the *insert_buffer* method. You can specify usage of these special single-pin cells to fix hold violations on the order of picoseconds, either alone or together with the *-buffer_list* option.

`-target_violation_type endpoint | tns`

Specifies the target violation type for clock network fixing. This option can be used only when *clock_network* is specified for the *-cell_type* option.

- *endpoint* (the default) - Timing fixing targets improvement of endpoint slack for each endpoint without degrading any existing endpoint violation.
- *tns* - Timing fixing targets improvement of overall Total Negative Slack (TNS). It might degrade existing endpoint violations even further, up to the current Worst Negative Slack (WNS) as long as TNS improves. If the *-wns_limit* option is used, it allows degradation of existing endpoint violations to the specified WNS limit.

Note: The *tns* value requires a PrimeTime-ADV license.

`-wns_limit limit_value`

Specifies the limit of Worst Negative Slack (WNS) for the *-target_violation_type tns* option. The default is the existing WNS.

`-unfixable_reasons_format text | csv`

Generates an unfixable reasons report file in the specified format, either *text* or *csv* (comma-separated values). The CSV report format is compatible with the PrimeTime GUI and spreadsheet programs. By default, without this option, both *text* and *csv* files are generated.

`-unfixable_reasons_prefix string`

Specifies a prefix used for naming the file generated by the *-unfixable_reasons_format* option. For example, specifying the prefix "abc" and using the CSV format writes out a file named abc_eco_tim.csv. If you omit this option, no files are generated.

Description

The *fix_eco_timing* command fixes or improves timing violations by sizing cells and inserting buffers or inverter pairs. It attempts to fix the specified types of violations while minimizing the impact on area and power.

After the violations are fixed, you can write out the design changes as a script by using the *write_changes* command. You can use the script to implement the same changes in another PrimeTime session or another tool (Design Compiler, IC Compiler, or IC Compiler II). A list of design changes created in this flow is called an engineering change order (ECO).

The *fix_eco_timing* command has options to specify the following fixing parameters:

- The type of timing violations to fix (setup or hold)
- The scope of the design to be fixed (from/to startpoints/endpoints, path groups, a path collection, or the whole design)
- The fixing methods (cell sizing, buffer insertion, or inverter pair insertion)
- The list of library cells that can be used for buffer or inverter pair insertion
- The types of cells to be modified (data paths, sequential cells, or clock networks)
- The targeted amount of timing margin (slack) to achieve
- The graph-based analysis mode (graph-based, path-based, or path-based exhaustive)
- The physical placement and sizing mode (none, open site, occupied site, or freeze silicon)

You must specify the fixing type, either *setup* or *hold*, using the *-type* option because of the different nature of these violations. By default, setup fixing uses cell sizing alone to reduce data path delays, whereas hold fixing uses both cell sizing and buffer insertion to increase data path delays. By default, fixing occurs only in data paths, not in clock networks.

The command performs fixing using multiple iterations, starting with the worst violations. It repeats fixing iterations until all violations are fixed or it determines that further fixing is not worth the runtime cost, based on the current quality of results and fixing option settings. You can generate a report on unfixable violations by using the *-verbose* or *-estimate_unfixable_reasons* option. Based on the report, you can target the unfixed violations using another *fix_eco_timing* command with different option settings.

Setup fixing seeks to avoid introducing design rule checking (DRC) violations, but it is allowed to introduce hold violations because setup violations are harder to fix. Hold fixing seeks to avoid introducing both setup and DRC violations. To fix both setup and hold violations, fix setup violations first, then hold violations, using separate *fix_eco_timing* commands.

The *fix_eco_timing* command is compatible with single-core analysis, distributed multi-scenario analysis (DMSA), and multicore analysis. To ensure consistent results during DMSA fixing, apply any *dont_touch*, *dont_use* and *size_only* attributes identically to all scenarios.

The *-from*, *-to*, *-path_selection_options*, and *-group* options allow targeted fixing of the design. The *-from* and *-to* options restrict fixing to paths with specific startpoints and endpoints. The *-path_selection_options* restricts fixing to a path collection created in a manner similar to the *get_timing_paths* command. The *-group* option restricts fixing to named path groups.

The *fix_eco_timing* command requires pin slack information. If the *timing_save_pin_arrival_and_slack* variable is not already set to *true*, the *fix_eco_timing* command automatically sets it to *true* and updates the design with arrival and slack information.

The *fix_eco_timing* command is intended for the signoff flow using delay and slew calculation in the PrimeTime tool, based on extracted parasitic data. If delays and slews are annotated directly (for example, in SDF format), the quality of results cannot be guaranteed.

Setup Fixing

By default, setup fixing uses cell sizing alone to reduce data path delays. You can specify the maximum area increase by setting the *eco_alternative_area_ratio_threshold* variable. By default, the variable is set to 2, which limits the area increase to twice the area of the cell before fixing. Note that this limit applies to *each iteration* of setup fixing, so in multiple fixing iterations, the final size can be more than twice the original size. If you set the variable to 0, there is no size limit on upsizing.

Limiting the area increase can result in better timing predictability after physical implementation of the ECO changes, as the changes are less likely to disturb the existing layout during incremental placement and routing. However, more changes (and more runtime) might be required to achieve the needed slack improvement.

In addition to cell sizing, you can use buffer insertion (*-methods {insert_buffer}*) to reduce path delays to critical loads. It is recommended that you perform cell sizing first, using a separate *fix_eco_timing* command, to reduce the number of buffers required and minimize the layout disturbance.

For setup fixing using buffer insertion, the *-physical_mode* option must be set to *open_site*, *occupied_site*, or *freeze_silicon*. The fixing rate depends very much on having empty sites available in the layout. Using the *-buffer_list* option, specify a good representative set of buffers that cover the necessary delay and area range. Be sure to specify buffers with good drive strength and short delays, and omit buffers with long delays.

Buffers and delay cells placed by the hold fixing tool might hurt setup paths. To remove the redundant buffers that cause the setup violation, use the *-methods {remove_buffer}* otion. A buffer in a setup violation path is removed if the following conditions are met:

- It does not worsen existing violations or does not cause the slack to become worse than the margins set by the *-hold_margin* or *-setup_margin* options.
- Removing the buffer does not introduce any new DRC violations: *max_transition*, *max_capacitance*, or *max_fanout*.

The *fix_eco_timing -methods {remove_buffer}* command only removes buffer cells. To also perform cell sizing for setup fixing, use a separate *fix_eco_timing* command with appropriate option settings.

Hold Fixing

By default, hold fixing uses both cell sizing and buffer insertion to increase delays in the data path (*-methods {size_cell insert_buffer}*). You can choose to use these methods separately or together. When used together, the command performs cell sizing first, then buffer insertion, to reduce the number of buffers required and minimize the layout disturbance. Cell sizing includes downsizing, upsizing, and same-size cell replacement of buffer and delay cells to fix hold violations.

When using the *-buffer_list* option, specify a representative set of buffers that cover the necessary delay range. However, avoid specifying an excessive number of similar buffers, which can lead to longer runtimes with no fixing benefit.

To fix small hold violations (on the order of 5 ps or less), you can insert load capacitance cells in the data path instead of buffers. To insert load cells, use the *fix_eco_timing -type hold* with the *-load_cell_list* option with a list of load cells. If you use both the *-load_cell_list* and *-buffer_list* options in the same *fix_eco_timing* command, the tool automatically fixes small hold violations by inserting load cells and larger hold violations by inserting buffers.

Restricting the Library Cells Used for Sizing

To prevent the tool from using specific library cells for sizing, apply the *dont_use* attribute to these library cells with the *set_dont_use* command. Note that this usage restriction applies only to cell sizing, not buffer insertion. The tool uses buffers specified by the *-buffer_list* option, irrespective of the *dont_use* attribute.

Usage of library cells can also be restricted by applying the *pt_dont_use* user-defined attribute on the restricted lib_cell objects, which is a legacy feature. For an example of how to apply this attribute, see the EXAMPLES section.

Preserving Specific Cells and Nets During Fixing

To prevent the *fix_eco_timing* command from performing fixing on specific cells or nets, apply a *dont_touch* attribute to the cell or net. For example,

```
pt_shell> set_dont_touch [get_nets n123] true
```

The *fix_eco_timing* command does not size any cell with a *dont_touch* setting and does not insert a buffer or inverter pair on any net with a *dont_touch* setting.

If you apply a *dont_touch* setting to a hierarchical cell, that setting propagates downward to the child cells and nets of the hierarchical cell. You can override the *dont_touch* setting for specific lower-level cells and nets by setting the *dont_touch* attribute to *false* for those objects.

Preventing Specific Cell Instances From Being Removed

Among the cells, you might want to prevent specific instances from being removed but allow other sizing operations on them. To do this, use the *set_size_only* command. For example,

```
set_size_only [get_cells {U28}] true
```

This command sets the *size_only* attribute to *true* for cell instance U28, which prevents the cell from being removed by ECO commands.

Fixing Margin

The *-setup_margin* and *-hold_margin* options let you specify overfixing or underfixing of timing violations. By default, when the command finds a violation, it tries to fix the violation to attain a slack of zero. Setting a positive margin changes the target to the specified positive slack value (overfixing); this results in safer timing but requires more fixing effort. Conversely, specifying a negative margin changes the target to the specified negative slack value (underfixing); this does not completely fix the violation but is more likely to be successful.

Note that the margin setting affects only the target slack for violations being fixed; it has no effect on the choice of paths to be fixed. To choose violations for fixing based on existing slack values, use the *-slack_lesser_than* and *-slack_greater_than* options.

You can specify both a setup margin and a hold margin for a fixing operation. This can be useful during hold fixing because downsizing cells or inserting buffers in a data path to fix a hold violation can worsen setup timing for the same path. By specifying a setup margin of zero or more during hold fixing, the tool checks for setup timing effects and ensures that no new setup violations are introduced.

When you use path-based analysis (*-pba_mode* option set to *path* or *exhaustive*), the margin setting applies to path-based calculated slack only for the matching fixing type, either setup or hold. For example, during hold fixing (*-type hold*), the *-hold_margin* setting applies to the path-based slack being recalculated, whereas setup checking performed in the background uses graph-based analysis, so the *-setup_margin* setting applies to the graph-based slack.

Total Percentage of Violations Fixed (Fix Rate)

Upon completion of fixing, the *fix_eco_timing* command reports the total percentage of violations fixed, sometimes called the "fix rate." For example,

```
Fixing Summary:
-----------------------------------------------
Total violating endpoints found              25
Total violating endpoints fixed              23
Total violating endpoints remaining           2
Total percentage of violations fixed      92.0%
```

This report is based on the number of violating endpoints fixed, which can be less than the number of fixed violations of various types. For example, when the command fixes rise and fall violations from multiple startpoints that end at a single endpoint, all of the fixed violations count as a single fixed violating endpoint.

In distributed multi-scenario analysis, the fix rate is based on the sum of all detected and fixed violating endpoints across all scenarios, even if some endpoints overlap between the scenarios. For example, suppose that scenario S1 has 100 violating endpoints and scenario S2 has and 200 violating endpoints. If the *fix_eco_timing* command fixes 99 violations in scenario S1 and 198 in scenario S2, the fix rate is (99+198)/(100+200) = 99.0%.

Remaining Violations

The *fix_eco_timing* command applies an intelligent adaptive algorithm to optimize runtime, memory, fix rate, and number of changes. It typically runs multiple fixing iterations in an attempt to address all violations. However, some violations can remain unfixed due to the high cost of fixing.

For example, if a path has a huge hold violation that would require 15 buffers to be inserted in one stage, the *fix_eco_timing* command stops fixing the path because this large number of buffers might cause a large timing difference in the physical implementation tool. In this case, it is recommended to fix such violations in the implementation tool or adjust the constraints so that violations are more realistic.

However, if you still want to fix these violations, you can run the *fix_eco_timing* command multiple times, perhaps using different fixing options, until they are all fixed.

Unfixable Violation Report

The *fix_eco_timing* command, upon completion of violation fixing, generates a report on unfixable violations when you use the *-verbose* option. You can get the same type of report without performing any actual fixing by using the *-estimate_unfixable_reasons* option.

To generate the report, the *eco_report_unfixed_reason_max_endpoints* variable must be set to a positive integer. For DMSA, this variable must be set in the manager.

Here is a report example:

```
pt_shell> fix_eco_timing -type hold -buffer_list {BUFX1 BUFX2 BUFX3} \\
          -physical_mode open_site -verbose
  ...
Unfixable violations:
  A - There are available library cells outside area limit
  B - Delay improvement is too small to fix the violation
  C - The violation is in clock network
  D - Cell or net is located in high density area
  E - Physical information is incomplete or unavailable
  H - Logical and physical hierarchies are inconsistent
  I - Buffer insertion with given library cells cannot fix the violation
  L - Available physical area limits the use of one or more library cells
  O - No open free site is available
  R - No locations are available in parasitics or location transformation
 failed
  S - Cell sizing with alternative library cells cannot fix the violation
  T - Timing margin is too tight to fix the violation
  U - UPF restricts fixing the violation
  V - Net or cell is invalid or has dont_touch attribute
  W - Fixing the violation might degrade DRC violations

  Violation                                              Reasons  Prio/slk
  ------------------------------------------------------------------------
  S:U2/Z                                                   T         P7
    U3/Z                                                 I W         P6
    U4/Z                                                   U         P9
    U6/Z                                                   U         P6
  E:U8/I                                                           -2.105
  C:U3/Z                                                 I W         P6
    U5/Z                                                   S         P3
    U7/Z                                                   U         P0
  E:U9/A                                                           -0.307
  ...
```

The "Violation" column shows the cell pins where unfixable violations are located. The letter codes S and E indicate the start and end of a sequence of pins in a violating path. The letter code C indicates continuation of a path segment from a point in a timing path with a shared segment already reported earlier in the table.

The "Reasons" column shows the reason or reasons that a violation could not be fixed at the cell pin, using letter codes from the key shown at the beginning of the report.

The "Prio/slk" column shows the fixing priority or endpoint slack. For each pin in a timing path, the fixing priority is ranked from P0 to P9 (lowest to highest priority), with the fixing of higher-priority pins potentially more effective in reducing violations than lower-priority pins. For each path endpoint ("E" in the Violation column), the total path slack is shown.

In the foregoing example, the first entry in the table is pin U2/Z, at the start (S) of the timing path having the worst slack. The unfixable reason "T" means that U2/Z was not

fixed because the timing margin was too tight, and fixing the timing at this point has a relatively high priority (P7).

The entries in the table are organized in order from path startpoint (S) to path endpoint (E), and the timing paths are shown in order of slack, worst-slack paths first. The first path sequence shows the full path from U2/Z through U8/I. At the path endpoint, the path slack is reported in the "Prio/slk" column, -2.105.

The second path sequence is reported as starting from U3/Z and ending at U9/A, omitting the segment of the path from U2/Z and U3/Z already reported for the first path. The letter C at the begriming of this sequence indicates the continuation of a path segment partially shared with an earlier path, starting with the last shared point, which helps to avoid duplicate entries in the table.

You can interpret the reported priority P0 through P9 as a fixing bottleneck ranking. If you are able to fix only some of a large number of violations, to get the best possible gain from each fix, you can collect stages with higher priorities and try to fix them first.

These are the unfixable reason codes reported in the "Reasons" column:

- *A - There are available library cells outside area limit*

  Library cells that might have been used to fix the violation exceeded the area limit specified by the *eco_alternative_area_ratio_threshold* variable. You might be able to fix the violation by increasing the area limit.

- *B - Delay improvement is too small to fix the violation*

  The violating slack was too negative to be fixed by improving the delay in this stage. You can improve (but not completely fix) the violation by setting a negative margin for the target slack using the *-setup_margin* or *-hold_margin* option.

- *C - The violation is in clock network*

  The violation is in a clock network. By default, the command does not attempt to fix timing violations in clock networks because doing so can change the clock tree timing and possibly create new timing violations. If data path fixing and sequential cell fixing are not successful (*-cell_type combinational*, *-cell_type sequential*), you can try clock network fixing (*-cell_type clock_network*).

- *D - Cell or net is located in high density area*

  There is not enough room physically available in the vicinity of the violation. In the physical implementation tool (such as IC Compiler II), check the area utilization in that location. Note that when the *eco_allow_filler_cells_as_open_sites* variable is set to *true* (the default), physically aware ECO commands treat filler cells as open sites.

- *E - Physical information is incomplete or unavailable*

The physical information is incomplete or unavailable for the specific net or cell, or the net is marked as "+USE CLOCK" in the DEF file specified by the *set_eco_options* command. See if the DEF and LEF files are consistent with your current design.

- *H - Logical and physical hierarchy is inconsistent*

  The netlist and physical layout are not consistent, and a buffer cannot be inserted. Use the physical implementation tool to resolve the inconsistency.

- *I - Buffer insertion with given library cells cannot fix the violation*

  The violation cannot be fixed using the library cells listed by the *-buffer_list* option. Specifying a wider range of buffer cells might allow the command to fix the violation.

- *L - Available physical area limits the use of one or more library cells*

  The area of a library cell that could be used for fixing exceeds the available area in the physical layout. Use the physical implementation tool to increase the available area.

- *M - Violations in leaf cells are less than the threshold*

  The tool did not perform a fix at this clock network location because the number of violations that could be fixed in the leaf cells of the transitive fanout of this location is less than the threshold specified by the *-clock_fixes_per_change violation_limit* option. The fix might be performed if you set a lower limit.

- *N - Cell level higher than the threshold*

  The tool did not perform a fix at this clock network location because the clock tree level of this location is higher than the threshold specified by the *-clock_max_level_from_reg* option. The fix might be performed if you set a higher limit or disable the limit.

- *O - No open free site is available*

  There is no empty free site available on the route or in the vicinity of the violation. Inspect the LEF/DEF file for filler cells and see whether they are treated as empty free sites. When the *eco_allow_filler_cells_as_open_sites* variable is set to *true* (the default), physically aware ECO commands treat all filler cells as empty free sites. Also, you can use the physical implementation tool (such as IC Compiler II) to create empty free sites.

  If the *-physical_lib_constraint_file* option of the *set_eco_options* command has been used to specify intercell or advanced-node spacing rule constraints, some empty sites might not be wide enough to be used. In that case, you can use the physical implementation tool to examine the spacing rule constraints and the cells in the vicinity of such free sites.

- *Q - Capture and launch timing conflict with each other*

Both the capture and launch sides of a flip-flop have timing violations, and the *-cell_type* option is set to *clock_network*. For example, suppose that a flip-flop has a setup slack of -2.0 on the D pin and a hold slack of -1.0 on the Q pin. Changing the clock arrival time to fix one violation would worsen the other violation, so the two violations cannot be fixed by this method. If a branch cell in a clock network shows this unfixed violation reason, it means that a leaf sequential cell in its clock tree has violations on both the capture and launch sides.

- *R - No locations are available in parasitics or location transformation failed*

  Either the parasitics files do not contain location coordinates, which are required to use the buffer insertion method for setup fixing, or location transformation has failed.

- *S - Cell sizing with alternative library cells cannot fix the violation*

  All alternative library cells have been considered for sizing to fix the violation, but the violation cannot be fixed by cell sizing. Use a buffer insertion method to fix the violation, if needed.

- *T - Timing margin is too tight to fix the violation*

  The violation is on a critical or near-critical setup or hold timing path and the opposite constraint does not have enough timing slack to allow fixing of the targeted violation. To allow fixing to proceed, you can relax the fixing constraints by specifying negative values for the *-setup_margin* and *-hold_margin* options.

- *U - UPF restricts fixing the violation*

  Certain UPF cells cannot be sized or buffered. For details, see the "UPF Cells" section of this man page.

- *V - Net or cell is invalid or has dont_touch attribute*

  One or more leaf cells or nets under the cell have the *dont_touch* attribute set to *true*. Check to see if the reason is justified.

- *W - Fixing the violations might degrade DRC violations*

  Timing fixing would create new DRC violations or worsen existing DRC violations. To fix timing violations at the expense of DRC violations, use the *-ignore_drc* option.

To write out the unfixable violation report as a separate file, use the *-unfixable_reasons_format* option and specify either *text* or *csv* as the format. To specify a prefix for the file name, use the *-unfixable_reasons_prefix* option.

The CSV report file is compatible with the PrimeTime GUI and spreadsheet programs. To read the report into the PrimeTime GUI, choose the Report > ECO Unfixed Violation menu command. You can use the PrimeTime GUI to investigate and fix the remaining violations.

The entries in the violation table are cross-referenced to the timing reports, path inspector, and layout view shown in the GUI.

UPF Cells

By default, UPF cells are not considered for sizing. UPF cells are cells that have one or more of the following attributes set to true: *always_on*, *is_isolation*, *is_retention*, or *is_level_shifter*.

To allow UPF cells to be sized, set the *eco_allow_sizing_with_lib_cell_attributes* variable. For example,

```
pt_shell> set_app_var eco_allow_sizing_with_lib_cell_attributes \
  {always_on is_isolation is_retention is_level_shifter}
```

For distributed multi-scenario analysis (DMSA), set this variable in each worker process.

A buffer inserted into a net must be placed in the same power domain as the driver and all loads of the net. Buffer insertion is not performed on the following nets:

- Nets connected to any pins of an always_on cell
- Nets connected to control and clock pins of a retention cell
- A net between a port and an isolation cell
- A net between a port and a level shifter cell

Note that the *fix_eco_drc* command allows always_on buffer insertion on nets connected with an always_on cell if the *eco_allow_insert_buffer_always_on_cells* variable is set to *true* (false by default).

Writing Changes

After you fix violations using the *fix_eco_timing* command and other ECO commands, you can write out the design changes as a script by using the *write_changes* command. You can use the script to implement the same changes in another PrimeTime session or another tool (Design Compiler, IC Compiler, or IC Compiler II). Use the *-format* option of the *write_changes* command to specify the format of the script.

The number of changes in the *write_changes* script can be different from the number of sized cells and inserted buffers reported by the *fix_eco_timing* command. For example, if the *fix_eco_timing* command inserts a buffer in the first iteration and then sizes the buffer in the next iteration, it reports two changes, one buffer insertion and one sizing. The *write_changes* command combines these two actions into a single change, insertion of the final sized buffer.

After you implement the changes in a physical implementation tool such as IC Compiler II, extract new parasitic data, write a new Verilog file, and check the final design timing again in the PrimeTime tool.

Multiple Fixing Scenarios With Fewer Hosts with Hybrid Timing View

In the hybrid timing view ECO flow, the tool uses a combination of live views for accuracy-critical ECO scenarios and static views of less critical scenarios. The static-view timing and violation information is merged into the live-view scenarios at intervals determined by the tool.

Suppose you have 50 scenarios for your fixing. If you run ordinary DMSA, you need to have 50 worker processes simultaneously running in one or multiple machines.

In the hybrid timing view ECO flow, if you have 15 live-view scenarios that cover 90 percent of the violations, the tool can merge the remaining 35 scenarios with the 15 live-view scenarios as static views. Then can run fixing with only 15 DMSA worker processes instead of 50. Note that when the live views have a relatively low coverage of violations (for example, 80 percent), you might not be able to achieve the desired fixing coverage. For more information, see the man page for the *fBwrite_eco_scenario_data* and the *fBstart_eco_scenarios* commands.

Ignoring DRC Violations

If you use the *-ignore_drc* option, the command ignores DRC violations as it tries to fix timing violations. This option is recommended only for later stages of an ECO cycle, after you have fixed all violations possible without causing or worsening DRC violations. When no more timing violations can be fixed due to DRC constraints, you can use this option to trade off DRC violations to fix more timing violations.

Physically Aware Timing Fixing

Set the *-physical_mode* option to *open_site* or *occupied_site* to enable the command to load physical data into memory and use that data while fixing timing violations. Before you run the *fix_eco_timing* command, use the *set_eco_options* command to specify the paths to the physical data, either IC Compiler II database files or LEF/DEF library and design data exchange files:

```
set_eco_options \\
  -physical_tech_lib_path LEF_tech_file_list \\
  -physical_lib_path LEF_lib_file_list \\
  -physical_design_path DEF_design_file_list

set_eco_options \\
  -physical_icc2_lib icc2_lib_directory_path \\
  -physical_icc2_blocks icc2_block_name_list
```

The *fix_eco_timing* command reads in the physical data files as specified by the *set_eco_options* command if they are not already read in and checked by the *check_eco* command. To view the chip layout in the GUI, open the GUI with the *gui_start* command and then choose Window > Layout Window.

With the physical data loaded into memory, the tool considers physical constraints such as available free space, cell density, and net topology when it sizes cells and inserts buffers. For cell sizing, the tool selects a library cell that meets both the timing and physical constraints. For buffer insertion, the tool searches for the best placement location to maximize timing improvement while minimizing disturbance of existing cells in the layout.

Early in the ECO cycle, to maximize the fix rate, you can set the *-physical_mode* option to *occupied_site*, which allows sized cells and inserted buffers to overlap existing cells, as long as those cells to be moved to make space for the changes. Late in the ECO cycle, to minimize layout disturbance, set the option to *open_site* mode, which preserves the placement of existing cells.

After you finish running the *fix_eco_timing* command, you can write a changelist file that can be used by the physical implementation tool to implement the ECO changes. With the physical mode enabled, the commands in the changelist file might contain placement information.

The following *add_buffer_on_route* command written by the *write_changes* command specifies the layout coordinates for the new buffer in the IC Compiler or IC Compiler II tool.

```
add_buffer_on_route net1 BUF1X -location {100.0 200.0 0}
```

The following *size_cell* command written by the *write_changes* command specifies the instance to be sized and the new library cell to be used, which fits in the available space. It does not specify any location because the cell is upsized in its current location.

```
size_cell U16 AND2X
```

The following *insert_buffer* command written by the *write_changes* command places a new buffer in an available free site at a specified location.

```
insert_buffer U2/Z BUF2X -location {200.0 300.0 0}
```

After physical data is loaded into memory, the PrimeTime tool keeps track of ECO changes and updates the physical database appropriately. Therefore, it is recommended that you perform only physically aware fixing in the whole PrimeTime session. For example, after you have performed fixing with the *-physical_mode* option set to *open_site* or *occupied_site*, do not run the *fix_eco_timing* command again with the option set to *none*. Otherwise, the logic-only fixing changes can corrupt the physical database, resulting in significant layout disturbance when the changes are made in the physical implementation tool. For the same reason, you should avoid what-if commands such as the *size_cell* and *insert_buffer* commands.

Physically Aware Freeze Silicon Flow

In the freeze silicon flow, fixing is performed only by buffer insertion using spare cells already implemented in silicon. Only the interconnect mask layers are modified, which saves the high cost of modifying the silicon layer masks. Thus, when the *-physical_mode*

option is set to *freeze_silicon*, the *-methods* option can be set only to *insert_buffer*. The freeze silicon mode works with the IC Compiler II physical implementation tool.

In this flow, timing fixing uses spare cells defined in the DEF file or IC Compiler II database. You must identify the spare cells by using the *set_eco_options* command with the *-programmable_spare_cell_names* option. You can query the current ECO option settings by using the *report_eco_options* command.

An ECO change in the freeze silicon mode inserts a buffer directly on top of a matching programmable spare cell. The *write_changes* command writes out the change in the changelist file as an *insert_buffer* or *add_buffer_on_route* command output with location. The exact location of the buffer in the programmable spare cell is guided by the layout and the net topology.

The *write_changes* command also writes out a *map_freeze_silicon* command corresponding to each buffer. The IC Compiler II tool uses this command to program the mapped buffer into the spare cell. If the mapped buffer uses only a portion of a programmable spare cell, the *map_freeze_silicon* command backfills the leftover space with one or more smaller matching programmable spare cells, so that the leftover space can be used later for implementing further ECO changes.

Fixing in Multiply Instantiated Modules (MIMs)

When you set the *eco_enable_mim* variable to *true* (the default is *false*), the tool performs the same ECO changes across each set of MIMs. In physically aware fixing, the tool recognizes a set of MIM instances when they share the same DEF file; in logic-only fixing, it recognizes them when they share the same parasitics file according to the *-path* option in the *read_parasitics* command.

Suppose a CPU module is instantiated four times in the TOP module as CPU1, CPU2, CPU3 and CPU4, and the files CPU.def and CPU.SPEF are associated with the CPU module. The tool derives the MIM configuration when the following settings are used.

In physically-aware fixing,

```
set_eco_options -physical_design_path {TOP.def CPU.def}
```

In logic-only fixing,

```
read_parasitics -path {CPU1 CPU2 CPU3 CPU4} CPU.SPEF
```

or

```
read_parasitics -path [all_instance CPU] CPU.SPEF
```

If MIMs are detected, the *fix_eco_timing* command shows the current MIM configuration before it starts fixing. Verify that the MIM configuration matches your expectations. If not, check the DEF file or parasitics file configuration in your setup script.

When fixing violations, a change in one MIM instances is replicated in all instances of the MIM set. For example, a buffer insertion in CPU1 is replicated in CPU2, CPU3, and CPU4. Whenever the tool fixes a violation, it analyzes all MIM instances and makes sure that it does not cause timing violations in other instances. Thus, it is possible to get a lower fix rate with MIM ECO enabled because timing might be more constrained by multiple instances.

After fixing is completed, use the *write_changes* command to generate the changelist file for the associated MIM instances. In the preceding example, one changelist file is written for the CPU1 through CPU4 instances. For more information, see the man page of the *write_changes* command.

Per-Scenario ECO Margins

In distributed multi-scenario analysis, you can use different setup and hold margins for different scenarios by specifying their values in the *-timing_setup_margin* and *-timing_hold_margin* options of the *set_eco_options* command in the worker scripts.

Note that the *-setup_margin* and *-hold_margin* options of the *fix_eco_timing* command have priority over the corresponding options of the *set_eco_options* command.

Clock Network Fixing

When the *-cell_type* option is set to *clock_network*, the command makes changes in clock networks to fix timing violations. Clock network fixing methods include cell upsizing, cell downsizing, buffer or inverter pair insertion and bypass.

If the *bypass_buffer* method is applied, new timing is annotated on the changed nets based on net topology, capacitance and net delay change estimates. For signoff accurate timing, it is recommended to run the *implement_eco* command to implement the changes and get new timing after extracting new parasitics from the changed nets.

The *fix_eco_timing* command has options to control the scope of the changes in the clock network:

- Use the *-clock_max_level_from_reg* option to limit the changes to lower levels of the clock trees. For example, setting the option to 1 limits changes to the leaf level of the clock tree.
- Use the *-clock_fixes_per_change* option to reduce the number of changes in the clock network by setting a minimum number of violations to fix per change. For example, setting the option to 4 requires at least four violations to be fixed per change.

In physically aware timing fixing, you must use the *set_eco_options* command with the *-physical_enable_clock_data* option before you use the *fix_eco_timing* command. You must specify valid buffer library cells that are available to the physical implementation tool, meeting any special rules such as nondefault routing (NDR) rules. The PrimeTime tool does not perform any extra rule checking for the provided buffers.

The PrimeTime tool uses all available alternative library cells to size cells in the clock network. If your design uses specialized library cells in the clock network, use a Tcl script similar to the following example to limit the sizing candidates to selected library cells. This example limits the library cells to those that have prefix "CLK" in their library cell base names.

```
define_user_attribute -type boolean -class lib_cell is_clk_cell
set clk_lcells [get_lib_cell */CLK*]
set_user_attribute $clk_lcells is_clk_cell true
set eco_alternative_cell_attribute_restrictions is_clk_cell
```

In an unfixable violation report generated by using the *-verbose* or *-estimate_unfixable_reasons* option, the unfixable reason codes *M*, *N*, and *Q* apply to clock network fixing, as described in the "Unfixable Violation Report" section of this man page.

The tool does not perform fixing on a clock mesh, where multiple drivers drive the same net. However, if the clock mesh is connected to a normal clock tree, the tool can perform ECO changes in the normal clock tree portion of the clock network.

TNS Driven Timing Fixing With Clock Network ECO

If you specify *tns* for the *-target_violation_type* option, the command attempts to improve overall Total Negative Slack (TNS) even if it degrades existing endpoint violations. However, the tool limits existing violation degradation to the current Worst Negative Slack (WNS). If you want to improve TNS further by allowing WNS degradation, use the *-wns_limit* option to specify the allowed limit.

Suppose a design has WNS of -100, and a specific flip-flop has slack -60 at D pin and -40 at Q pin. The command may degrade slack from -60 to -100 at D pin if TNS improves. If -80 is specified for the *-wns_limit* option, however, the command only degrades slack from -60 to -80 to meet the limit requirement. If -200 is specified for the *-wns_limit* option, the command can degrade slack from -60 to -200 as long as TNS improves.

Leakage Power

Leakage power can increase during timing fixing. By default, timing fixing performs optimization to meet timing requirements while minimizing the increase in area, without considering leakage.

If leakage power is more important than area, you can use the *-power_attribute* option to optimize power during timing fixing, based on a library cell attribute that defines the leakage power value for the cell. For example,

```
# Create a user-defined attribute "leak_attr" for lib_cell object
define_user_attribute leak_attr -type float -class lib_cell

# Assign values that reflect leakage of each lib_cell at worst corner
set_user_attr -class lib_cell [get_lib_cell LIB_H/INV1X_HVT] leak_attr
 1.0
```

```
set_user_attr -class lib_cell [get_lib_cell LIB_H/INV2X_HVT] leak_attr
 2.0
set_user_attr -class lib_cell [get_lib_cell LIB_L/INV1X_LVT] leak_attr
 10.0
set_user_attr -class lib_cell [get_lib_cell LIB_L/INV2X_LVT] leak_attr
 15.0
 ...

# Perform setup timing fixing with leakage power consideration
fix_eco_timing -type setup -power_attribute leak_attr

# Perform hold timing fixing with leakage power consideration
fix_eco_timing -type hold -power_attribute leak_attr
```

While fixing timing violations, the tool chooses library cells to minimize the increase in leakage power, and does not consider area. It replaces only the existing cells that have the leak_attr attribute, and replaces them only with other cells that also have the leak_attr attribute.

Instead of assigning the leakage values explicitly, you can import the attribute values from the cell library database. For details, see SolvNet 2371111, "Extracting Leakage Power for Library Cells"; and SolvNet 2040404, "PrimeTime J-2014.12 Update Training," Module 8: ECO - Power Recovery.

In distributed multi-scenario analysis, the respective library cells in different libraries must be assigned matching leakage power values.

PrimePower Dynamic, Leakage, and Total Power

The PrimePower tool performs comprehensive power analysis using actual switching activity and library-defined power data such as dynamic and leakage power of each cell. The tool performs a power analysis when you use the *update_power* or *report_power* command at the pt_shell prompt.

Total power (both dynamic and leakage) can increase when cells are upsized to improve timing. By default, the command performs optimization to fix timing violations while minimizing the increase in area, without considering power.

To better manage the total power overhead, you can use PrimePower power analysis data in timing fixing by using the *-power_mode* option set to *dynamic*, *leakage*, or *total*. Then timing fixing tries to reduce the overhead of the dynamic (switching) power, leakage power, or total (dynamic plus leakage) power as measured by the *report_power* command while fixing similar timing violations. For hold fixing, only the *leakage* setting is supported.

With the *-power_mode total* option, the tool chooses actions to fix setup or hold timing violations while reducing the total power overhead. The action chosen can vary depending on the local conditions in the design. For example, it can choose to fix setup timing violations by either upsizing the cell or by lowering the cell threshold voltage, the choice depending on the local switching activity. High switching activity favors lowering the

threshold voltage (to reduce switching power overhead), whereas low switching activity favors upsizing the cell (to reduce leakage power overhead). A similar strategy applies to buffer insertion. Among the buffers that give similar fix, the ones with lower total power overhead are favored.

With the *-power_mode leakage* option, the tool fixes setup or hold timing violations considering leakage power without requiring user-defined power attributes.

In the distributed multi-scenario analysis (DMSA) flow, the tool gets its dynamic power data from exactly one scenario, which you specify with the *-dynamic_scenario* option. Similarly, it gets its leakage power data from exactly one scenario, which you specify with the *-leakage_scenario* option. These options are used only in a DMSA flow. In general, you should specify the scenario showing the worst dynamic power and worst leakage power, respectively, for these two options. They could be the same scenario or two different scenarios.

Load Capacitance Cells for Hold Fixing

Hold fixing can use load capacitance (dummy load) cells along the data path to target violations on order of picoseconds using the *insert_buffer* method. You can use the *-load_cell_list* option separately from or together with the *-buffer_list* option.

When the fixing method is specified as {*size_cell insert_buffer*} with both the *-load_cell_list* and *-buffer_list* options, the command performs cell sizing first, then load capacitance cell insertion, followed by buffer insertion. This sequence minimizes the layoutdisturbance. With the *-physical_mode open_site* option, to control the search area for inserting load capacitance cells, set the *eco_insert_buffer_search_distance_in_site_rows* variable.

The *-load_cell_list* option can use buffer cells, inverter cells, and dedicated load cells for load cell insertion. Buffer and inverter cells inserted as load cells have unconnected outputs. Dedicated load cells are smaller single-pin cells that are easier to place. They have the following attributes:

```
Attribute Name      Type          Value
-----------------------------------------
function_id         string        unknown
is_black_box        boolean       true
is_combinational    boolean       true
number_of_pins      int           1
```

The file written by the *write_changes -format icctcl* command contains commands like the following to set the location and orientation of each new load cell:

```
create_cell U_LOAD_CAP_CELL_1 CLOAD1
connect_net $existing_target_net [get_pins {U_LOAD_CAP_CELL_1/A}]
set_cell_location -coordinates {150.00 30.60} -orientation N
```

Feedback

Examples

The following example overfixes setup violations with a slack worse than -0.1 to achieve a positive setup slack of 0.1 (while ignoring paths with a setup slack better than -0.1):

```
pt_shell> fix_eco_timing -type setup -slack_lesser_than -0.1
 -setup_margin 0.1
```

The following example overfixes setup violations with a slack worse than 0.2 to achieve a positive setup slack of 0.2. Note that this overfixes some positive-slack paths, for example, by increasing the slack from 0.1 to 0.2.

```
pt_shell> fix_eco_timing -type setup -slack_lesser_than 0.2 -setup_margin
 0.2
```

The following example overfixes all setup violations to achieve a positive setup slack of 0.2, while preserving a hold slack of at least 0.1:

```
pt_shell> fix_eco_timing -type setup -setup_margin 0.2 -hold_margin 0.1
```

The following example fixes setup violations with slack between -2.0 and 0.0 (while ignoring paths with violations worse than -2.0):

```
pt_shell> fix_eco_timing -type setup -slack_greater_than -2.0
```

The following example fixes hold violations using only cell sizing.

```
pt_shell> fix_eco_timing -type hold -methods size_cell
```

The following example fixes hold violations using only pin buffering.

```
pt_shell> fix_eco_timing -type hold \\
            -methods {insert_buffer_at_load_pins
 insert_buffer_at_driver_pins} \\
            -buffer_list {BUFX1 BUFX2 BUFX3}
```

The following example fixes a small set of hold violations that have an exhaustive path-based slack of less than zero:

```
pt_shell> fix_eco_timing -type hold -pba_mode exhaustive \\
           -buffer_list {BUFX2 DLY1X2 DLY2X2}
```

The following example fixes setup violations only in a selected set of path groups, with verbose reporting of unfixable violations:

```
pt_shell> fix_eco_timing -type setup -group {SYSCLK* IOCLK} -verbose
```

The following example fixes setup violations only by sizing sequential cells:

```
pt_shell> fix_eco_timing -type setup -cell_type sequential
```

The following example fixes setup violations by sizing both sequential and combinational cells:

```
pt_shell> fix_eco_timing -type setup -cell_type {sequential
 combinational} \\
 -methods size_cell
```

The following example applies the *pt_dont_use* user-defined attribute to a set of library cells. You must define the user-defined attribute first.

```
pt_shell> define_user_attribute pt_dont_use \\
          -quiet -type boolean -class lib_cell
```

You can use the attribute directly with the *set_user_attribute* command, or for convenience, you can define the following procedure:

```
proc set_pt_dont_use {lib_cell} {
  set_user_attribute \\
    -class lib_cell \\
    [get_lib_cell -quiet $lib_cell] \\
    pt_dont_use true
  }
```

Now you can use this procedure to apply the attribute:

```
set_pt_dont_use {lib/CLKBUF* lib/CLKMUX*}
```

In the distributed multi-scenario analysis (DMSA) flow, you must define and apply the attribute at the scenarios using the *remote_execute* command.

The following example uses the *-physical_mode* option to specify the *open_site* mode.

```
pt_shell> fix_eco_timing -type hold -physical_mode open_site \\
           -buffer_list {BUFX2 DLY1X2 DLY2X2}
```

The following example uses the *-path_selection_options* option to target specific paths for fixing setup violations.

```
pt_shell> fix_eco_timing -type setup -path_selection_options \\
           {-through u01/A -to ff0/D -max_paths 10 -nworst 5}
```

The following example uses the *-path_selection_options* option to target specific paths for fixing hold violations.

```
pt_shell> fix_eco_timing -type hold -buffer_list {BUFX2 DLY1X2 DLY2X2}
 \\
           -path_selection_options {-delay_type min -from ff0/Q \\
           -max_paths 100 -nworst 10}
```

In the distributed multi-scenario analysis flow, you must define variables in the worker processes if you want to use them in the *-path_selection_options* option. The following

example uses the *-path_selection_options* option in distributed multi-scenario analysis with a variable.

```
pt_shell> remote_exec {set startpoints [get_pins {ff0/Q ff1/Q}] }
pt_shell> fix_eco_timing -type hold -buffer_list {BUFX2 DLY1X2 DLY2X2}
 \\
           -path_selection_options {-delay_type min -from $startpoints
 \\
           -max_paths 100 -nworst 5}
```

Alternatively, you can define the variable as a list instead of a collection:

```
pt_shell> remote_exec {set startpoints "{ff0/Q ff1/Q}"}
pt_shell> fix_eco_timing -type hold -buffer_list {BUFX2 DLY1X2 DLY2X2}
 \\
           -path_selection_options {-delay_type min -from $startpoints
 \\
           -max_paths 100 -nworst 5}
```

The following example performs setup fixing by modifying the clock network.

```
pt_shell> fix_eco_timing -type setup -cell_type clock_network
```

The following example uses the *insert_inverter_pair* method to perform setup fixing in a clock network.

```
pt_shell> fix_eco_timing -type setup -methods insert_inverter_pair \\
           -cell_type clock_network -buffer_list {INVX1 INVX2 INVX4}
```

The following example uses the *insert_inverter_pair* method to perform hold fixing in data paths.

```
pt_shell> fix_eco_timing -type hold -methods insert_inverter_pair \\
           -buffer_list {INVX1 INVX2 INVX4}
```

The following example uses the *-estimate_unfixable_reasons* option to generate a report of unfixable reasons without actually fixing the design. The variable *eco_report_unfixed_reason_max_endpoints* must be set to a positive integer.

In this example, if the command reports unfixable reason code "I: buffer list with given lib cells cannot fix violation," you can modify the buffer list to include a larger range of buffer library cells and get a quick estimate of the results before you start actual fixing.

```
pt_shell> set_app_var eco_report_unfixed_reason_max_endpoints 50
pt_shell> fix_eco_timing -type hold -buffer_list {BUFFX1 BUFFX2} \\
           -estimate_unfixable_reasons
```

The following example uses the *load_cell_list* option to perform hold fixing for violations in order of picoseconds along data path.

Feedback

```
pt_shell> fix_eco_timing -type hold -methods insert_buffer \\
           -load_cell_list {CLOAD1 CLOAD2} -slack_greater_than -0.005
```

The following example uses the *-load_cell_list* option in single operation along with the *-buffer_list* option to fix hold violations.

```
pt_shell> fix_eco_timing -type hold -methods insert_buffer \\
           -load_cell_list {CLOAD1 CLOAD2} -buffer_list {BUFX1 BUFX2}
```

The following example uses the *remove_buffer* method to fix setup violations.

```
pt_shell> fix_eco_timing -type setup -methods remove_buffer
```

See Also

- create_clock
- get_timing_paths
- group_path
- insert_buffer
- remote_execute
- report_eco_options
- report_eco_scenarios
- report_timing
- set_dont_use
- set_dont_touch
- set_size_only
- set_eco_options
- set_user_attribute
- start_eco_scenarios
- write_changes
- write_eco_scenario_data
- eco_allow_filler_cells_as_open_sites
- eco_allow_insert_buffer_always_on_cells
- eco_allow_sizing_with_lib_cell_attributes

Feedback

- eco_alternative_area_ratio_threshold
- eco_enable_mim
- eco_report_unfixed_reason_max_endpoints
- timing_save_pin_arrival_and_slack

---

fix_eco_wire

Performs ECO wire optimization by spacing, widening, rerouting wires, or changing layers.

Syntax

status *fix_eco_wire*

```
[-methods method_list]
[-path_selection_options option_string]
[-nets net_list]
[-min_routing_layer layer_name]
[-max_routing_layer layer_name]
```

Data Types

```
method_list       list
option_string     string
net_list          list
layer_name        string
```

Arguments

`-methods method_list`

Specifies the fixing methods, one or more of *space_wire*, *widen_wire*, *reroute_wire*, or *change_layer*. The default is *-methods {space_wire widen_wire change_layer}*.

`-path_selection_options option_string`

Specifies the path selection options for the *get_timing_paths* command, which the *fix_eco_wire* command uses to collect timing paths for fixing. Based on the specified methods, *fix_eco_wire* identifies fixable target nets under collected timing paths automatically. Because fixing is performed only on these paths, to check the quality of results after fixing, use the *get_timing_paths* or *report_timing* commands with the same options.

The -path_selection_options and -nets options are mutually exclusive; you can specify only one.

`-nets` *net_list*

Specifies the list of target nets for ECO wire optimization. To check the quality of results after fixing, use the *get_timing_paths* or *report_timing* commands with -nets options.

The -nets and -path_selection_options options are mutually exclusive; you can specify only one.

`-min_routing_layer` *layer_name*

Specifies the minimum routing layer for the nets. Only one layer can be specified. The *-methods {change_layer}* fixing method is required for the -min_routing_layer option. If -min_routing_layer is not specified, the tool automatically assigns it to the lowest low resistance layer when it is available. If low resistance layers are not available in the design, -methods {change_layer} will be ignored.

`-max_routing_layer` *layer_name*

Specifies the maximum routing layer for the nets. Only one layer can be specified. The *-methods {change_layer}* fixing method is required for the -max_routing_layer option. If -max_routing_layer is not specified, the tool automatically assigns it to the highest low resistance layer when it is available. If low resistance layers are not available in the design, -methods {change_layer} will be ignored.

Description

The *fix_eco_wire* command performs ECO wire optimization for automatically or manually specified nets. This command is available for PrimeECO where it performs timing and routing co-optimization for better QoR with less routing disturbance.

ECO wire optimization includes spacing, widening, rerouting wires, and changing layers, which are enabled by space_wire, widen_wire, reroute_wire, and change_layer methods, respectively. For specified methods, *fix_eco_wire* applies relevant non-default routing to the target nets and reflects the timing effect of wire fixing for users to check the quality of results after fixing. Note that *set_routing_rule* applies the specified non-default routing to the target nets but does not change the timing. ECO wire optimization changes performed by *set_routing_rule* are classified as "manual" method.

After you finish running the *fix_eco_wire* command, you can implement the ECO changes with *implement_eco* command.

Requirements

The minimum requirements to run the *fix_eco_wire* command is identical to those for *implement_eco* command.

Feedback

Examples

The following example uses the *-path_selection_options* option to target specific paths for spacing and rerouting wire fixing.

```
prompt> fix_eco_wire -methods {space_wire reroute_wire} \\
           -path_selection_options {-max_paths 20 -nworst 1}
```

The following example uses the *-nets* option to target specific nets for widening and layer-changing wire fixing.

```
prompt> fix_eco_wire -nets {widen_wire change_layer} \\
           -nets {n43, n134} -min_routing_layer M5 -max_routing_layer M6
```

See Also

- reset_eco_wire
- report_eco_wire
- get_timing_paths
- report_timing
- implement_eco

---

foreach_in_collection

Iterates over the elements of a collection.

Syntax

string *foreach_in_collection*

```
itr_var
collections
body
```

Data Types

```
itr_var             string
collections         list
body                string
```

Arguments

itr_var

Specifies the name of the iterator variable.

collections

Specifies a list of collections over which to iterate.

body

Specifies a script to execute per iteration.

Description

The *foreach_in_collection* command is used to iterate over each element in a collection. You cannot use the Tcl-supplied *foreach* command to iterate over collections because the *foreach* command requires a list, and a collection is not a list. Also, using the *foreach* command on a collection causes the collection to be deleted.

The arguments for the *foreach_in_collection* command parallel those of the *foreach* command: an iterator variable, the collections over which to iterate, and the script to apply at each iteration. All arguments are required.

Note: The *foreach_in_collection* command does not allow a list of iterator variables.

During each iteration, the *itr_var* option is set to a collection of exactly one object. Any command that accepts *collections* as an argument also accepts *itr_var* because they are of the same data type (collection).

You can nest the *foreach_in_collection* command within other control structures, including another *foreach_in_collection* command.

Note that if the body of the iteration is modifying the netlist, it is possible that all or part of the collection involved in the iteration will be deleted. The *foreach_in_collection* command is safe for such operations. If a command in the body causes the collection to be removed, at the next iteration, the iteration ends with a message indicating that the iteration ended prematurely.

An alternative to collection iteration is to use complex filtering to create a collection that includes only the desired elements, then apply one or more commands to that collection. If the order of operations does not matter, the following are equivalent. The first is an example without iterators.

```
set s [get_cells {U1/*}]
command1 $s
command2 $s
unset s
```

The following is a similar approach using the *foreach_in_collection* command:

```
foreach_in_collection itr [get_cells {U1/*}] {
  command1 $itr
  command2 $itr
}
```

For collections with large numbers of objects, the non-iterator version is more efficient, though both produce the same results if the commands are order-independent.

Feedback

Examples

The following example from PrimeTime removes the wire load model from all hierarchical cells in the current instance.

```
pt_shell> foreach_in_collection itr [get_cells *] {
?             if {[get_attribute $itr is_hierarchical] == "true"} {
?               remove_wire_load_model $itr
?             }
?          }
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
```

See Also

- collections

---

g

---

gen_ctpm

Generate the CTPM for a library with different target spice model.

Syntax

string *gen_ctpm*

```
-base_lib base_lib
-side_file side_file
-header header_file
-subckt subckt_folder
-simulator sim_exec
[-max_cores number]
[-temperature temperature]
[-sub_ext extention]
[-sub_prefix prefix]
[-lib_cells lib_cell_name_list]
[-output output_name]
[-verbose]
```

Data Types

```
base_lib            string
side_file           string
header_file         string
subckt              string
sim_exec            string
number              int
temperature         float
extention           string
```

```
prefix              sting
lib_cell_name_list  list
output_name         string
```

Arguments

`-base_lib` *base_lib*

Specifies the base library, which is characterized using original spice model.

`-side_file` *side_file*

Specifies the sensitivity augmented library which is the side-file of base library, as used in *define_sensitivity_lib_mapping base_lib -side_file side_file*.

`-header eader_file`

Specifies the target spice header file, all needed spice options needs to be included.

`-subckt` *subckt_folder*

Specifies the target spice subckt folder contains all lib_cells' subckt files with name as {prefix}{lib_cell_name}{extention}. You can use a single subckt file without -sub_ext and -sub_prefix, but runtime will be significantly slower.

`-simulator` *sim_exec*

Specifies the HSPICE simulator path used for simulation.

`-max_cores` *number*

Specifies the upper bound of number of cores used for the command on local machine. Default value is 16.

`-temperature` *temperature*

Specifies the retargeting temperature, if it's different from base library.

`-sub_ext` *extention*

Specifies the subckt file extention for per lib_cell subckt in *subckt_folder*. Default is ".spi".

`-sub_prefix` *prefix*

Specifies the subckt file prefix for per lib_cell subckt in *subckt_folder*. Default is "".

`-lib_cells` *lib_cell_name_list*

Specifies the list of lib_cells to run gen_ctpm. By default, all lib_cells in library will be included. This option is used for flow pipe-clean purpose to generate CTPM for only a single lib_cell or specific lib_cells in the library.

`-output` *output_name*

Specifies the output name of CTPM. By default, the generated CTPM name is base_lib db file name and replace ".db" with ".ctpm". Example, base_lib file is "mylib1.db", default generated CTPM name is "mylib1.ctpm".

`-verbose`

Generates the lib_timinig_arc based correlation details for each arc. The correlation data is printed for rise and fall seperately and contains following details: 1) Relative and absolute error for pre- and post-CTPM, for both delay and slew 2) The pass/fail column indicates whether the post-CTPM error crosses the threshold defined by *set_check_ctpm_tolerance* command. By default it's 3%/3ps.

Description

The *gen_ctpm* command can be used to generate the compact timing power model (CTPM) of any target spice model. By generating the CTPMs, the PrimeShield tool enables the timing and power impact assessment of a given target. You can define the target using a SPICE model representing the next process design kit (PDK), the spot model, or a different SPICE corner. The CTPM generation in the PrimeShield tool uses an advanced machine learning algorithm. The aim is to bridge the gap between the base and the target by capturing the gap through the SPICE process parameters placed in a CTPM database. The generated CTPMs are used in a base STA session. When a CTPM is used in a base STA session, the quality of results (QoR) of the session improves to match the QoR of the target. You can perform this targeting analysis without the characterized libraries for that target, thereby saving time on cycles waiting for official characterization libraries. This targeting analysis only requires the augmented libraries representing the timing behavior of each arc, for a given physical SPICE parameter perturbation range.

Examples

The following script is generating CTPM with target_header, where subckt per lib_cell in subckt_dir are named as "{lib_cell_name}{.spi}".

ps_shell> gen_ctpm \\ -base_lib base.db \\ -side_file augmented_sensitivity.db \\ -header target_header.spi \\ -subckt subckt_dir/ \\ -simulator /global/apps/hspice_2022.06-1/hspice/bin/hspice \\ -max_cores 16

```
Information: Spice2Design ML-CTPM saved at base.ctpm (PSCTPM-027)
Statistical Report for rise:
SUMMARY for all rise lib_cell_num= 571 lib_timing_arc_num= 14109
run_name avg std 99-th_percentile
delay-preCTPM  -6.14% 5.14% 18.08%
delay-postCTPM   0.03% 2.89% 6.74%
slew-preCTPM  -3.40% 2.59% 9.43%
slew-postCTPM   0.10% 1.67% 4.00%
Statistical Report for fall:
SUMMARY for all fall lib_cell_num= 605 lib_timing_arc_num= 14062
```

Feedback

```
run_name avg std 99-th_percentile
delay-preCTPM  -6.37% 1.93% 10.85%
delay-postCTPM  -0.05% 1.31% 3.09%
slew-preCTPM  -3.93% 2.45% 9.63%
slew-postCTPM  -0.01% 1.12% 2.62%
```

See Also

- read_ctpm
- report_ctpm
- ps_enable_spice2design_analysis

---

generate_power_profile

Generate power profile.

Syntax

status *generate_power_profile*

file_name]

Data Types

file_name	string

Arguments

ile_name

Specifies the name of the file for power profile generation.

Description

This command is used to generate power profile. The power profile can be either static or time based. Static power profile uses csv format, while time based profile uses xlsx format.

For power profile generation, user need to provide LEF/DEF files through the command *set_eco_options* with options *-physical_tech_lib_path -physical_lib_path -physical_design_path*.

For static power profile generation, static power analysis has be run first before invoking the command *generate_power_profile*. For time based power profile generation, user has to invoke *update_power* first for time based power analysis. Since the power profile generation is based on the waveform data in FSDB file from *update_power* command, the following options have to be specified before *update_power*: *set_power_analysis_option -include all_with_leaf -separate_power_waveform all*. This is to store leaf cell based waveform data including all power components in the FSDB file. The FSDB file name is

obtained from the command *set_power_analysis_option -waveform_output*. The default FSDB file name is primetime_px.fsdb.

Examples

The following example generates static power profile.

```
pt_shell> update_power
pt_shell> set_eco_options -physical_tech_lib_path tech.lef
pt_shell> set_eco_options -physical_lib_path design.lef
pt_shell> set_eco_options -physical_design_path design.def
pt_shell> generate_power_profile power.csv
```

The following example sets power analysis option to include power waveform data on all leaf cells for all power components. Then it invokes time based power analysis, and generates time based power profile.

```
pt_shell> set_app_var power_analysis_mode time_based
pt_shell> read_vcd test.vcd
pt_shell> set_power_analysis_option -include all_with_leaf
 -separate_power_waveform all
pt_shell> update_power
pt_shell> set_eco_options -physical_tech_lib_path tech.lef
pt_shell> set_eco_options -physical_lib_path design.lef
pt_shell> set_eco_options -physical_design_path design.def
pt_shell> set_power_profile_options -row 100 -column 100
 -enable_time_based true
pt_shell> generate_power_profile power.xlsx
```

See Also

- set_power_profile_options

---

get_alternative_lib_cells

Creates a collection of library cells that can be used to replace a cell instance in the current design using the *size_cell* command.

Syntax

```
collection get_alternative_lib_cells
  [-current_library]
  [-libraries lib_spec]
  [-base_names]
  cell
```

Data Types

```
lib_spec      list
cell          string
```

Feedback

Arguments

`-current_library`

Restricts selection of the new library cells to only the same library as that of the cell being considered for replacement. You can use the this option only if the *lib_cell* argument specifies a cell instance or a base cell name without a library name.

`-libraries` *`lib_spec`*

Specifies a list of libraries from which to select alternative library cells. You can specify either a list of library names or a collection created by the *get_libs* command. The default is to select from all libraries specified in the *link_path* variable.

`-base_names`

Returns a list of library cell base names, without library names. If library cells from different libraries share the same base name, the duplicate names are removed from the result. This option is enabled by default if you invoke pt_shell with the *-multi_scenario* option.

`cell`

Specifies a single cell instance in the design or full library cell name. The command creates a collection of alternative library cells that are functionally equivalent to the specified cell instance or library cell.

Description

The *get_alternative_lib_cells* command creates a collection of library cells that are functionally equivalent to a specified cell or library cell, but can have different properties such as size and drive strength. You can relink an instance of the cell to an alternative library cell by using the *size_cell* command.

A library cell is included in the collection only if it satisfies the same criteria used by the *size_cell* command to determine whether a library cell is equivalent. Cells are equivalent if they have the same:

- Number of input pins
- Number of output pins
- Logical functionality
- Number, type, and sense of enabled cell timing arcs

By default, the I/O pins do *not* need to have the same names or ordering. Furthermore, a 2-input NAND gate and a 2-input OR gate, each cell with one negated input, are functionally equivalent, by DeMorgan's theorem.

To enforce identical pin naming, set the *eco_strict_pin_name_equivalence* variable to true. In that case, the command considers matching pin names and their respective functions as an additional requirement for equivalence.

To disable the requirement that timing arcs must match between the alternative library cell and the original library cell, set the *eco_strict_lib_arc_equivalence* variable to *false*.

To see a comparison of the slack and design cost within a set of alternative library cells, use the *report_alternative_lib_cells* command.

Examples

The following example shows how to find alternative library cells for a cell instance in the design. Here, U1_max is an instance of the library cell lib_max/OR3 and U2_min is an instance of lib_min/OR3.

```
pt_shell> set link_path [list * lib_max.db]
* lib_max.db
pt_shell> set_min_library lib_max.db -min_version lib_min.db
Loading db file '/u/libs/lib_max.db'
Loading db file '/u/libs/lib_min.db'
Created max/min library relationship:
  Max: /u/libs/lib_max.db:lib_max
  Min: /u/libs/lib_min.db:lib_min
 ...
pt_shell> get_alternative_lib_cells -libraries lib_min U1_max
{"lib_min/OR3", "lib_min/OR3P"}
pt_shell> get_alternative_lib_cells -libraries lib_max U2_min
{"lib_max/OR3", "lib_max/OR3P"}
pt_shell> get_alternative_lib_cells U1_max
{"lib_max/OR3P", "lib_min/OR3", "lib_min/OR3P"}
pt_shell> get_alternative_lib_cells U2_min
{"lib_max/OR3", "lib_max/OR3P", "lib_min/OR3P"}
pt_shell> get_alternative_lib_cells U2_min -current_library
{"lib_min/OR3P"}
pt_shell> get_alternative_lib_cells -base_names U2_min
{"OR3", "OR3P"}
```

The following example shows how to find alternative library cells for a lib_cell object.

```
pt_shell> get_alternative_lib_cells lib_max/OR3
{"lib_max/OR3P", "lib_min/OR3", "lib_min/OR3P"}
pt_shell> get_alternative_lib_cells lib_max/OR3 -current_library
{"lib_max/OR3P"}
```

See Also

- get_libs
- report_alternative_lib_cells

Feedback

- set_min_library
- size_cell
- link_path
- link_path_per_instance

---

get_app_var

Gets the value of an application variable.

Syntax

string *get_app_var*

```
[-default | -details | -list]
[-only_changed_vars]
var
```

Data Types

```
var     string
```

Arguments

`-default`

Gets the default value.

`-details`

Gets additional variable information.

`-list`

Returns a list of variables matching the pattern. When this option is used, then the *var* argument is interpreted as a pattern instead of a variable name.

`-only_changed_vars`

Returns only the variables matching the pattern that are not set to their default values, when specified with *-list*.

var

Specifies the application variable to get.

Description

The *get_app_var* command returns the value of an application variable.

There are four legal forms for this command:

- `get_app_var <var>`

Returns the current value of the variable.

- `get_app_var <var> -default`

Returns the default value of the variable.

- `get_app_var <var> -details`

Returns more detailed information about the variable. See below for details.

- `get_app_var -list [-only_changed_vars] <pattern>`

Returns a list of variables matching the pattern. If *-only_changed_vars* is specified, then only variables that are changed from their default values are returned.

In all cases, if the specified variable is not an application variable, then a Tcl error is returned, unless the application variable *sh_allow_tcl_with_set_app_var* is set to true. See the *sh_allow_tcl_with_set_app_var* man page for details.

When *-details* is specified, the return value is a Tcl list that is suitable as input to the Tcl *array set* command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key, and each even-numbered element in the list is the value of the previous key.

The supported keys are as follows:

name

This key contains the name of the variable. This key is always present.

value

This key contains the current value of the variable. This key is always present.

default

This key contains the default value of the variable. This key is always present.

help

This key contains the help string for the variable. This key is always present, but sometimes the value is empty.

type

This key contains the type of the application variable. Legal values of for this key are: string, bool, int, real. This key is always present.

constraint

This key describes additional constraints placed on this variable. Legal values for this key are: none, list, range. This key is always present.

min

This key contains the min value of the application variable. This key is present if the constraint is range. The value of this key may be the empty string, in which case the variable only has a max value constraint.

max

This key contains the max value of the application variable. This key is present if the constraint is "range". The value of this key may be the empty string, in which case the variable only has a min value constraint.

list

This key contains the list of legal values for the application variable. This key is present if the constraint is "list".

Examples

The following are examples of the *get_app_var* command:

```
prompt> get_app_var sh_enable_page_mode
1

prompt> get_app_var sh_enable_page_mode -default
false

foreach {key val} [get_app_var sh_enable_page_mode -details] {   echo
 "$key: $val"
}
=>  name: sh_enable_page_mode
    value: 1
    default: false
    help: Displays long reports one page at a time
    type: bool
    constraint: none

prompt> get_app_var -list sh_*message
sh_new_variable_message
```

See Also

- report_app_var
- set_app_var
- write_app_var

Feedback

get_attribute

Retrieves the value of an attribute on an object or on a collection of objects.

Syntax

string *get_attribute*

```
[-class class_name]
[-quiet]
[-value_list]
-objects object_list
-name attribute_name
object_spec
attr_name
```

Data Types

```
class_name        string
object_spec       collection
attr_name         string
```

Arguments

`-class class_name`

Specifies the class name of the *object_spec* argument, provided the *object_spec* option is a name. The valid values for the *object_spec* option are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, *lib_pin*, and *clock*.

`-quiet`

Prevents the reporting of messages related to the nonexistence of an object, the nonexistence of an attribute on an object, or the unavailabilty of an attribute on the specified object class. However, more serious usage issues are reported, even when this option is specified.

`-value_list`

Indicates that the return value should be a list, even if there is only a single object specified to retrieve the attribute. Normally, the return value of the *get_attribute* command is a string if there is only a single object, and a list if multiple objects are specified.

`-objects object_list`

Specifies a list of objects from which to get the attribute values. Each element in the list is a collection of objects. This option is mutually exclusive with *object_spec*. Either one (and only one) of *-objects* or *object_spec* must be specified.

`-name attribute_name`

Specifies the name of the attribute whose value is returned. This option is mutually exclusive with *attr_name*. Either one (and only one) of *-name* or *attr_name* must be specified.

object_spec

Specifies an object from which to retrieve the attribute value. The *object_spec* argument must be either a collection of one or more objects, or a name which is combined with the *-class* option to find the object. If the *object_spec* option is a name, you must also use the *-class* option. This positional option is provided for compatibility with other tools.

attr_name

Specifies the name of the attribute where the value is retrieved. This positional option is provided for compatibility with other tools.

Description

This command retrieves the value of an attribute on an object or on a collection of objects. The object is either a string name of exactly one object, or a collection of one or more objects. The return value is a string or a list of strings if multiple objects are specified to retrieve an attribute.

Examples

In the following example, the first command defines the X attribute for cells. The second command sets the attribute to a value on all cells in this level of the hierarchy. The third command retrieves the value from one cell, combines it with the *full_name* application attribute, and creates a simple report.

```
pt_shell> define_user_attribute -type int -class cell X
pt_shell> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'
pt_shell> foreach_in_collection sel [get_cells *] {
            echo -n "On '[get_attribute $sel full_name]', "
            echo "X =  [get_attribute $sel X]"
          }
On 'i1', X =  30
On 'i2', X =  30
```

In the next example, the *get_attribute* command is used to retrieve a list of attribute values for a collection of cells.

```
pt_shell> set flipflops [get_cells ff*]
{"ffa", "ffb", "ffc", "ffd"}
pt_shell> get_attribute $flipflops is_sequential
true true true true
```

Feedback

See Also

- collections
- define_user_attribute
- foreach_in_collection
- get_defined_attributes
- list_attributes
- remove_user_attribute
- report_attribute
- set_user_attribute

---

get_cell

Creates a collection of cells from the current design relative to the current instance. You can assign these cells to a variable or pass them into another command.

Syntax

collection *get_cells*

```
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
[-of_objects objects]
patterns
```

Data Types

expression	string
objects	list
patterns	list

Arguments

`-hierarchical`

Searches for cells level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX *find* command. For example, if there is a cell block1/adder, a hierarchical search finds it using "adder".

`-filter` *`expression`*

Filters the collection with the *expression* value. For any cells that match *patterns* or *objects*, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

`-quiet`

Suppresses warning and error messages if there are no objects that match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive; you can specify only one.

`-of_objects` *`objects`*

Creates a collection of cells connected to the specified objects. In this case, each object is either a named pin, a pin collection, a named net, a net collection, a named library cell, or a collection of library cells. Note that library cells can not be combined with other types of objects. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

`patterns`

Matches cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type cell. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

Description

This command creates a collection of cells in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any cells match the *patterns* or *objects* option and passes the filter (if specified). If no objects match the criteria, an empty string is returned.

If any *patterns* or *objects* option fails to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_cells* command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the *query_objects* command. In addition, you can assign the *get_cells* result to a variable.

When issued from the command prompt, the *get_cells* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_cells* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command (for example, if you want to display the object class), use the *get_cells* command as an argument to the *query_objects* command.

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

Examples

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is not. The output is just a display.

```
pt_shell> get_cells "o*" -filter "ref_name == FD2"
{"o_reg1", "o_reg2", "o_reg3", "o_reg4"}
```

The following example shows that, given a collection of pins, you can query the cells connected to those pins.

```
pt_shell> set pinsel [get_pins o*/CP]
{"o_reg1/CP", "o_reg2/CP"}

pt_shell> query_objects [get_cells -of_objects $pinsel]
{"o_reg1", "o_reg2"}
```

The following example removes the wire load model from cells i1 and i2.

```
pt_shell> remove_wire_load_model [get_cells {i1 i2}]
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
1
```

Feedback

See Also

- collections
- filter_collection
- get_pins
- link_design
- query_objects
- collection_result_display_limit

---

get_cells

Creates a collection of cells from the current design relative to the current instance. You can assign these cells to a variable or pass them into another command.

Syntax

collection *get_cells*

```
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
[-of_objects objects]
patterns
```

Data Types

expression	string
objects	list
patterns	list

Arguments

`-hierarchical`

Searches for cells level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX *find* command. For example, if there is a cell block1/adder, a hierarchical search finds it using "adder".

`-filter expression`

Filters the collection with the *expression* value. For any cells that match *patterns* or *objects*, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.

Feedback

`-quiet`

Suppresses warning and error messages if there are no objects that match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive; you can specify only one.

`-of_objects` *`objects`*

Creates a collection of cells connected to the specified objects. In this case, each object is either a named pin, a pin collection, a named net, a net collection, a named library cell, or a collection of library cells. Note that library cells can not be combined with other types of objects. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

`patterns`

Matches cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type cell. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

Description

This command creates a collection of cells in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any cells match the *patterns* or *objects* option and passes the filter (if specified). If no objects match the criteria, an empty string is returned.

If any *patterns* or *objects* option fails to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always

anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_cells* command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the *query_objects* command. In addition, you can assign the *get_cells* result to a variable.

When issued from the command prompt, the *get_cells* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_cells* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command (for example, if you want to display the object class), use the *get_cells* command as an argument to the *query_objects* command.

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

Examples

The following example queries the cells that begin with "o" and reference an FD2 library cell. Although the output looks like a list, it is not. The output is just a display.

```
pt_shell> get_cells "o*" -filter "ref_name == FD2"
{"o_reg1", "o_reg2", "o_reg3", "o_reg4"}
```

The following example shows that, given a collection of pins, you can query the cells connected to those pins.

```
pt_shell> set pinsel [get_pins o*/CP]
{"o_reg1/CP", "o_reg2/CP"}

pt_shell> query_objects [get_cells -of_objects $pinsel]
{"o_reg1", "o_reg2"}
```

The following example removes the wire load model from cells i1 and i2.

```
pt_shell> remove_wire_load_model [get_cells {i1 i2}]
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
1
```

See Also

- collections
- filter_collection
- get_pins

Feedback

- link_design
- query_objects
- collection_result_display_limit

---

get_clock_crossing_points

Gets a collection of clock-crossing endpoints between the given launch and the capture clocks.

Syntax

string *get_clock_crossing_points*

```
[-from from_clock]
[-to to_clock]
[-valid]
[-quiet]
```

Data Types

```
from_clock    collection of one object
to_clock      collection of one object
valid         Boolean
quiet         Boolean
```

Arguments

`-from from_clock`

Launch clock for getting endpoints.

`-to to_clock`

Capture clock for getting endpoints.

`-valid`

Does not include endpoints that only have false paths reaching them.

`-quiet`

Does not issue an error if no endpoints are found.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Gets a collection of endpoints where the launch clock is the *from_clock* and the capture clock is *to_clock*. By default, all such endpoints are included, even though the paths between launch clock and capture clock are all *false*, or if there is a clock groups relationship between the clocks.

Feedback

To obtain only non-false path endpoints, use the *-valid* option. To not report any error if no such endpoints could be found, use the *-quiet* option.

Examples

The following example shows a simple usage:

```
pt_shell> get_clock_crossing_points -from CLK1 -to CLK2
{"ffa/D", "ffb/D"}
```

See Also

- report_clock_crossing

---

get_clock_network_objects

Returns a collection of objects that belong or relate to the direct clock network.

Syntax

collection *get_clock_network_objects*

```
-type cell | register | net | pin | clock_gating_output
[-include_clock_gating_network]
[clock_list]
```

Data Types

```
clock_list     list
```

Arguments

```
-type cell | register | net | pin | clock_gating_output
```

Specifies one of the following types of clock network objects to return:

- *cell* - Cells that belong to the clock network, including non-inverting or inverting buffers, combinational cells, library defined clock gating cells, etc.
- *register* - Latches, flip-flops or black-box IPs, such as memory cells, that are driven by the clock network.
- *net* - Nets that connect the clock network cells to other clock network cells, to the driven registers, or to the I/O ports of design.
- *pin* - Pins and ports that are connected with the clock network nets. Note that the clock pins of the driven registers are included.
- *clock_gating_output* - Output pins of clock gating cells, either defined by the library, inserted by Power Compiler, automatically inferred by PrimeTime,

or manually set by the user. The results are consistent with those of *report_clock_gating_checks*, and can be affected by other clock gating check related commands or variables.

`-include_clock_gating_network`

Indicates that clock gating networks are included in the clock network.

`clock_list`

Specifies the clock domains of which the clock network objects are returned. By default, the command returns all clock network objects.

Description

This command returns a collection of clock network objects of certain type (specified by the *object_type* option) that belong or relate to one or several clock domains (specified by the *clock_list* option), or all clock domains if the *clock_list* option is not specified.

A clock network is a special logic part of the design that propagates the clocks from the clock sources to the clock pins of latches, flip-flops (that function as anything but propagating clocks) or black-box IPs. The propagation also stops at design output ports, dangling pins or nets, or the sources of other clocks. The *get_clock_network_objects* command retrieves certain types of objects from the direct clock network (including the latches, flip-flops and black-box IPs driven by the clock network). If the specified clock is a generated clock, the propagation does not go backward to the clock network of its master clock. If the specified clock is a master clock, the propagation does not go forward to the clock network of its generated clocks either.

The *-include_clock_gating_network* option indicates that discrete logic structure functioning as clock gating is taken as belonging to the clock network. Only the typical clock gating logic is considered as qualified clock gating network to be included in the clock network. The typical clock gating logic starts from the output of a level sensitive latch driven by the specified clock, possibly goes through a couple of buffers, and comes back to the specified clock network through one of the input pins of an AND or OR gate. The input pin must be a PrimeTime clock check enable pin, either inferred or manually set. Therefore, similarly, the results can be affected by clock gating check related commands or variables. When the clock gating network is included in the clock network, all objects in the clock gating network is considered as clock network objects. The latch is regarded as a clock network *cell*, but not a clock network *register*. The *clock_gating_output* object type is not affected by this option.

Examples

The following command returns a collection of clock network pins of all clock domains in the design, not including pins of the clock gating networks.

```
pt_shell> get_clock_network_objects -type pin
```

Feedback

See Also

- create_clock
- create_generated_clock
- get_clocks
- get_generated_clocks
- remove_clock
- remove_clock_gating_check
- remove_disable_clock_gating_check
- remove_generated_clock
- report_clock
- report_clock_gating_check
- set_clock_gating_check
- set_disable_clock_gating_check
- timing_disable_clock_gating_checks

---

get_clock_relationship

Shows the relationship between two clocks.

Syntax

string *get_clock_relationship*

```
[-type logically_exclusive | physically_exclusive | asynchronous |
 allow_paths]
clock_list
```

Data Types

clock_list	list

Feedback

Arguments

`-type logically_exclusive | physically_exclusive | asynchronous | allow_paths`

Checks one of the following types of clock relationship:

- *logically_exclusive* - Returns true if two given clocks are logically exclusive.
- *physically_exclusive* - Returns true if two given clocks are physically exclusive.
- *asynchronous* - Returns true if two given clocks are asynchronous to each other.
- *allow_paths* - Returns true if two given clocks are asynchronous and the timing analysis is allowed between these two clocks.

clock_list

This required argument specifies a list of clocks, which must contain two entries.

Description

Shows the relationship between two given clocks. If the option *-type* is not specified, the command returns a list of all relationships that exist between the two given clocks.

If the option *-type* is specified, the command returns either *true* or *false* depending on the existence of the relationship of given type between the two clocks.

Examples

The following example defines two asynchronous clock domains. The relationship between two clocks is then shown:

```
pt_shell> set_clock_groups -asynchronous -name g1 -group CLK33 -group
 CLK50
1
pt_shell> get_clock_relationship {CLK33 CLK50}
{asynchronous}
pt_shell> get_clock_relationship {CLK33 CLK50} -type logically_exclusive
false
pt_shell> get_clock_relationship {CLK33 CLK50} -type asynchronous
true
```

See Also

- create_clock
- remove_clock_groups

Feedback

- report_clock
- set_clock_groups

---

get_clocks

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

Syntax

collection *get_clocks*

```
[-quiet]
[-regexp]
[-nocase]
[-filter expression]
patterns
```

Data Types

expression	string
patterns	list

Arguments

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-filter expression`

Filters the collection with the *expression* option. For any clocks that match the *patterns* argument, the expression is evaluated based on the clock's attributes. If the expression evaluates to true, the clock is included in the result.

patterns

Matches clock names against patterns. Patterns can include the wildcard characters "*" and "?".

Feedback

Description

The *get_clocks* command creates a collection of clocks in the current design that match certain criteria. The command returns a collection if any clocks match the *patterns* argument and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

You can use the *get_clocks* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_clocks* command result to a variable.

When issued from the command prompt, the *get_clocks* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_clocks* command provides a fast, simple way to display clocks in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use the *get_clocks* command as an argument to the *query_objects* option.

For information about collections and the querying of objects, see the *collections* man page. In addition, see the man page for the *all_clocks* command, which also creates a collection of clocks.

Examples

The following example applies the *set_max_time_borrow* command to all clocks in the design matching "PHI*".

```
pt_shell> set_max_time_borrow 0 [get_clocks "PHI*"]
```

The following example sets the variable clock_list to a collection of clocks that have period of 15.

```
pt_shell> set clock_list [get_clocks * -filter "period==15"]
```

See Also

- all_clocks
- collections
- create_clock
- query_objects
- report_clock
- collection_result_display_limit

Feedback

get_command_hooks

Get registered hooks for this command

Syntax

string *get_command_hooks commandName*

```
string commandName
```

Arguments

commandName

Command to query

Description

The *get_command_hooks* command returns information about any hooks registered on the specified command. The data returned is in Tcl dict format and may be queried using that command.

Examples

When a command has both before and after hooks registered the data will look like this

```
prompt> get_command_hooks place_opt
before {before0 report_timing}
after {after0 {echo "Done with placement"}}
```

See Also

- add_command_hook
- remove_command_hook
- get_current_hook_command

get_command_option_values

Queries current or default option values.

Syntax

get_command_option_values

[-default | -current]

-command *command_name*

Feedback

Data Types

command_name string

Arguments

-default

Gets the default option values, if available.

-current

Gets the current option values, if available.

-command *command_name*

Gets the option values for this command.

Description

This command attempts to query a default or current value for each option (of the command) that has default and/or current-value-tracking enabled. Details of how the option value is queried depend on whether one of the *-current* or *-default* options is specified (see below).

A "Tcl array set compatible" (possibly empty) list of option names and values is returned as the Tcl result. The even-numbered entries in the list are the names of options that were enabled for default-value-tracking or current-value-tracking and had at least one of these values set to a not-undefined value). Each odd-numbered entry in the list is the default or current value of the option name preceding it in the list.

Any options that were not enabled for either default-value-tracking nor current-value-tracking are omitted from the output list. Similarly, options that were enabled for default-value-tracking or current-value-tracking, but for which no (not-undefined) default or current value is set, are omitted from the result list.

If neither *-current* nor *-default* is specified, then for each command option that has either default-value-tracking or current-value-tracking (or both) enabled, the value returned is as follows:

- The current value is returned if current-value-tracking is enabled and a (not-undefined) current value has been set;
- Otherwise the default value is returned if default-value-tracking is enabled and a (not-undefined) default value has been set;
- Otherwise the name and value pair for the option is not included in the result list.

If *-current* is specified, the value returned for an option is the current value if current-value-tracking is enabled, and a (not-undefined) current value has been set; otherwise the name and value pair for the option is omitted from the result list.

Feedback

If *-default* is specified, the value returned for an option is the default value if default-value-tracking is enabled, and a (not-undefined) default value has been set; otherwise the name and value pair for the option is omitted from the result list.

The result list from *get_command_option_values* includes option values of both dash options and positional options (assuming that both kinds of options of a command have been enabled for value-tracking).

The command issues a Tcl error in a variety of situations, such as if an invalid command name was passed in with *-command*.

Examples

The following example shows the use of *get_command_option_values*:

```
prompt> test -opt1 10 -opt2 20
1

prompt> get_command_option_values -command test
-bar1 10 -bar2 20
```

See Also

- preview
- set_command_option_value

---

get_coupling_capacitors

Creates a collection of parasitic coupling capacitors from the specified net. You can assign these coupling capacitors to a variable or pass them into another command.

Syntax

collection *get_coupling_capacitors*

```
[-filter expression]
[-quiet]
[-parasitic_corners corner_list]
[-all_parasitic_corners]
[-of_objects objects]
[-from node_name]
[-to node_name]
[-unit]
[-info]
```

Data Types

```
expression          string
corner_list         list
```

```
objects             list
node_name           string
```

Arguments

`-filter` *expression*

Filters the collection with the *expression* value. For any coupling capacitors that match *patterns* or *objects*, the expression is evaluated based on the coupling capacitor's attributes. If the expression evaluates to true, the coupling capacitor is included in the result.

`-quiet`

Suppresses warning and error messages if there are no objects that match. Syntax error messages are not suppressed.

`-parasitic_corners` *corner_list*

Indicates the parasitic corner ordering for coupling capacitance values. This option is only applicable for parasitics information obtained using *read_parasitics* with the GPD parasitic format. If the *-parasitic_corners* option is not specified, then the single parasitic corner specified by *set parasitic_corner_name* is returned.

`-all_parasitic_corners`

Specifies that all parasitic corners are to be returned. The parasitics ordering is determined according to the corner definitions of the GPD parasitics.

`-of_objects` *objects*

Creates a collection of coupling capacitors associated with the specified objects. Each object must be a named net or net collection.

You must use either the *-of_objects* option or the *-from* and *-to* options.

`-from` *node_name*

Constrains the selected coupling_capacitors to be part of a path starting from the specified node. The node name can be either a pin, port, or an internal node, where the internal node name syntax is *net_name*:*node_id*, where the legal *node_id* range is from 1 to the number of nodes in the net. This option is used only with the *-to* option.

`-to` *node_name*

Constrains the selected coupling_capacitors to be part of a path ending with the specified node, using the same node name syntax as the *-from* option.

`-unit`

Show unit of coupling_capacitors

Feedback

`-info`

Get unit info of coupling_capacitors

Description

This command creates a collection of coupling capacitors for the specified nets that match certain criteria. You must use either the *-of_objects* option and specify a net collection or the *-from* and *-to* options to specify a path from which to get the coupling capacitors. If no objects match the criteria, an empty string is returned.

When issued from the command prompt, the *get_coupling_capacitors* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

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

Examples

The following example queries the parasitic coupling capacitors of net n22, subject to those objects whose max corner capacitance is greater than 0.5e-3.

```
pt_shell> get_coupling_capacitors -of_objects n22 \\
  -filter "capacitance_max > 0.5e-3"
{"coupling_capacitor"}
```

See Also

- collections
- filter_collection
- get_ground_capacitors
- get_resistors
- link_design
- query_objects
- collection_result_display_limit

---

get_current_hook_command

Get the currently running command string

Syntax

string *get_current_hook_command*

Feedback

Description

The *get_current_hook_command* command may only be run from within a command hook. The command returns the command string used to invoke the command.

Note that option values may have been abbreviated when the hooked command was executed. In that case the information returned will also include abbreviated option values.

See Also

- add_command_hook
- remove_command_hook
- get_command_hooks

---

get_current_power_domain

Gets the power domains that are included in the power analysis.

This command works only in UPF mode.

Syntax

string *get_current_power_domain*

Arguments

None.

Description

The *set_current_power_domain* command provides the control of calculating power consumption for the domains of interest. Domain-based power consumption data can be generated for a multipower domain design. Only the power dissipated in the blocks covered by the current power domain list is calculated and reported. The *get_current_power_domain* command returns the power domains included in the power analysis.

Examples

The following example shows generating domain-based power analysis results. The design has two subblocks, InstA and InstB, which are covered by power domains PD1 and PD2, respectively. The first *report_power* command generates the power analysis results for the whole design, which is the default behavior. The second *report_power* command generates the power analysis results for the power consumed in power domain PD1. The third *report_power* command generates the results for the power consumed in power domain PD2.

```
pt_shell> ...
pt_shell> create_power_domain PD1 -elements {InstA}
pt_shell> create_power_domain PD2 -elements {InstB}
pt_shell> ...
pt_shell> report_power
pt_shell> set_current_power_domain PD1
pt_shell> report_power
pt_shell> get_current_power_domain
pt_shell> set_current_power_domain PD2
pt_shell> report_power
pt_shell> get_current_power_domain
```

See Also

- set_current_power_domain

---

get_current_power_net

Gets the power nets that are included in the power analysis.

This command works in UPF mode only.

Syntax

string *get_current_power_net*

Arguments

None.

Description

The *set_current_power_net* command provides control of calculating power consumption for the power nets of interest. Power net-based power consumption data can be generated for a multi-supply design. Only the power dissipated on the cell or part of cell that is supplied by the current power netlist is calculated and reported. The *get_current_power_net* command returns the power nets being included in the power analysis.

Examples

The following example shows generating power net-based power analysis results. The design has two subblocks "InstA" and "InstB", which is covered by power domain PD1 and PD2, respectively. There are a total of six supply nets defined. Among them, four are power nets and two are ground nets. The primary power net for power domain "PD1" is "VDDIS", and the primary power net for power domain "PD2" is "VDDGS". The first *report_power* command generates the power analysis results for the whole design, which is the default behavior. The second *report_power* command generates the power consumption associated with power net "VDDI". The third *report_power* command

generates the power consumption associated with power net "VDDIS", which is the output of power switch "top_header". The fourth and fifth reports are for "VDDG" and "VDDGS", respectively.

```
pt_shell> ...
pt_shell> create_power_domain PD1 -elements {InstA}
pt_shell> create_power_domain PD2 -elements {InstB}
pt_shell> create_supply_net VDDI  -domain PD1
pt_shell> create_supply_net VDDIS -domain PD1
pt_shell> create_supply_net VSS   -domain PD1
pt_shell> set_domain_supply_net PD1 -primary_power_net VDDIS
 -primary_ground_net VSS
pt_shell> connect_supply_net -ports InstA/LS_u1/VDDL VDDI
pt_shell> connect_supply_net -ports InstA/LS_u1/VDD VDDIS
pt_shell> connect_supply_net -ports InstA/LS_u1/VSS VSS
pt_shell> create_power_switch top_header \\
                     -domain PD1 \\
                     -output_supply_port {NSLEEPOUT VDDIS} \\
                     -input_supply_port {NSLEEPIN VDDI} \\
                     -on_state {state1 NSLEEPIN {CNTL}} \\
                     -control_port { CNTL ctrl }
pt_shell> ...
pt_shell> create_supply_net VDDG  -domain PD2
pt_shell> create_supply_net VDDGS -domain PD2
pt_shell> create_supply_net VSS   -domain PD2 -reuse
pt_shell> set_domain_supply_net PD2 -primary_power_net VDDGS
 -primary_ground_net VSS
pt_shell> create_power_switch gprs_header \\
                     -domain PD2 \\
                     -output_supply_port {NSLEEPOUT VDDGS} \\
                     -input_supply_port {NSLEEPIN VDDG} \\
                     -on_state {state1 NSLEEPIN {CNTL}} \\
                     -control_port { CNTL ctrl }
pt_shell> ...
pt_shell> report_power
pt_shell> set_current_power_net { VDDI }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDIS }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDG }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDGS }
pt_shell> report_power
pt_shell> get_current_power_net
```

See Also

- set_current_power_net

Feedback

get_current_sms_scenario

Queries the SMS scenarios pushed by the closest/most recent *push_sms_scenario* command.

Syntax

string *get_current_sms_scenario*

Description

The *get_current_sms_scenario* command can be used to query the SMS scenarios pushed by the closest/most recent *push_sms_scenario* command. These are the SMS scenarios applicable in the current context. This command is only available with SMVA/SMC analysis.

Examples

The following example specifies 3 voltage reference names {high,low,typ} for each one of the supply groups SN1 and SN2. PrimeTime automatically considers all the relevant combinations of voltage levels, in this case up to nine prior to the use of the *push_sms_scenario* command. The example shows a sequence of a *push_sms_scenario* command followed by a *pop_sms_scenario* command and their expected effect on subsequent *get_current_sms_scenario* commands. For more examples, please look at the *push_sms_scenario* command manual.

```
pt_shell> set_voltage_levels SN1 -reference_names { high low typ }
pt_shell> set_voltage -o SN1 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN1 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN1 -reference_name typ 0.9 -min 1.0
pt_shell> set_voltage_levels SN2 -reference_names { high low typ }
pt_shell> set_voltage -o SN2 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN2 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN2 -reference_name typ 0.9 -min 1.0
```

The following command shows no outstanding SMS scenarios are pushed by any previous *push_sms_scenario* command.

```
pt_shell> get_current_sms_scenario
0
```

The following command pushes the SN1:low SMS scenario collection. A subsequent command *get_current_sms_scenario* shows the SN1:low SMS scenario collection is now applicable to commands in the current context.

```
pt_shell> push_sms_scenario -sms_scenarios [get_sms_scenario
 -supply_group SN1 -reference_name low]

pt_shell> get_attribute [get_current_sms_scenario] description
{SN1:low}
```

Feedback

The following command pops the SN1:low SMS scenario collection, ending its corresponding scope. There are no outstanding pushed SMS scenario collections after this command, as shown using the *get_current_sms_scenario* command.

```
pt_shell> pop_sms_scenario

pt_shell> get_current_sms_scenario
0
```

See Also

- push_sms_scenario
- pop_sms_scenario

---

get_defined_attributes

Returns attribute names defined for the specified class.

Syntax

string *get_defined_attributes*

```
-class class_name
-return_classes
[-details]
[-application]
[-user]
[attr_pattern]
```

Data Types

attr_pattern	string
class_name	string

Arguments

`-class class_name`

Specifies the class for which to get the attributes. Valid classes are application-defined. This option cannot be used with the *-return_classes* option.

`-return_classes`

Returns the set of application-defined classes. This option cannot be used with the *-class* option.

`-details`

Returns additional information on a single attribute.

Feedback

`-application`

Returns application-defined attributes only. This option is mutually exclusive with the *-user* option.

`-user`

Returns user-defined attributes only. This option is mutually exclusive with the *-application* option.

`attr_pattern`

Specifies a list of attribute names or glob-style patterns to check on the existences of.

Description

This command returns a list of attribute names that are defined for the specified class.

There are three legals forms for this command:

- *get_defined_attributess -return_classes*

  Returns the application defined object classes.

- *get_defined_attributes -class cell [<pattern>]*

  Returns all the attributes, optionally those matching a pattern.

- *get_defined_attributes -class cell -details <attrName>*

  Returns more detailed information about the attribute. See below for details.

When *-details* is specified, the return value is a Tcl list that is suitable as input to the Tcl *array set* command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key, and each even-numbered element in the list is the value of the previous key.

The supported keys are as follows:

name

This key contains the name of the attribute

info

The short help defined for the attribute

type

The type of value the attribute stores.

settable

Indicates that the value is settable with the *set_attribute* command.

application

The value is 1 if the attribute is application defined, else 0.

subscripted

The value is 1 if the attribute is subscripted, else 0.

min_value

Not always present. If present, the value is the minimum allowed value.

max_value

Not always present. If present, the value is the maximum allowed value.

allowed_values

Not always present. If present, the value is the set of allowed values.

allowed_subscripts

Present if the attribute is subscripted. Contains the list of legal subcripts for the attribute if the attribute uses global subscripts. If empty, then the subscripts vary per object.

smart_subscripts

Present if the attribute is subscripted. Contains the list of supported smart subscripts if the attribute supports smart subscripts.

collection_types

Present if the attribute is a collection type. Specifies the class names of objects that may be present.

default_subscript

Present on subscripted attributes if specification of the subscript value is optional.

user_derived

The value is 1 if the attribute was created by the *define_derived_user_attribute* command, else 0.

tcl_get_command

Present if the attribute is user_derived. Contains the command used to retrieve the attribute value.

tcl_set_command

Present if the attribute is user_derived. Contains the command used to set the attribute value

Feedback

tcl_remove_command

Present if the attribute is user_derived. Contains the command used to remove the attribute value.

tcl_subscript_command

Present if the attribute is user_derived and subscripted. Contains the command used to retrieve the valid subscripts for an object.

Examples

The following are examples of the *get_defined_attributes* command:

```
prompt> get_defined_attributes -class cell
name full_name object_class ....

prompt> get_defined_attributes -class cell *name*
name full_name reference_name ...

prompt> get_defined_attributes -class cell full_name -details
name full_name info {} type string application 1 settable 0
subscripted 0 user_derived 0
```

See Also

- get_attribute
- help_attributes

---

get_defined_commands

Get information on defined commands and groups.

Syntax

string *get_defined_commands* [-details]

[-groups] [*pattern*]

```
string pattern
```

Arguments

`-details`

Get detailed information on specific command or group.

`-groups`

Search groups rather than commands

Feedback

pattern

Return commands or groups matching pattern. The default value of this argument is "*".

Description

The *get_defined_commands* gets information about defined commands and command groups. By default the command returns a list of commands that match the specified pattern.

When *-details* is specified, the return value is a list that is suitable as input to the *array set* command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key name, and each even-numbered element in the list os the value of the previous key. The *-details* option is only legal if the pattern matches exactly one command or group.

When *-group* is specified with *-details*, the supported keys are as follows:

name

This key contains the name of the group.

info

This key contains the short help for the group.

commands

This key contains the commands in the group

When *-details* is used with a command, the supported keys are as follows:

name

This key contains the name of the command.

info

This key contains the short help for the command.

groups

This key contains the group names that this command belongs to.

options

This key contains the options defined for the command. The value is a list.

return

This key contains the return type for the command.

Feedback

Each element in the *options* list also follows the key value pattern. The set of available keys for options are as follows:

name

This key contains the name of the option.

info

This key contains the short help for the option.

value_info

This key contains the short help string for the value

type

The type of the option.

required

The value will be 0 or 1 depending if the option is optional or required.

is_list

Will be 1 if the option requires a list.

list_length

If a list contains the list length constraint. One of any, even, odd, non_empty or a number of elements.

allowed_values

The allowed values if the option has specified allowed values.

min_value

The minimum allowed value if the option has specified one.

max_value

The maximum allowed value if the option has specified one.

Examples

```
prompt> get_defined_commands *collection
add_to_collection append_to_collection copy_collection filter_collection
foreach_in_collection index_collection sort_collection
prompt> get_defined_commands -details sort_collection
name sort_collection info {Create a sorted copy of the collection}
groups {} options {{name -descending info {Sort in descending order}
value_info {} type boolean required 1 is_list 0} {name -dictionary info
{Sort strings dictionary order.} value_info {} type boolean required 1
is_list 0} {name collection info {Collection to sort} value_info
collection type string required 0 is_list 0} {name criteria info {Sort
```

Feedback

```
criteria - list of attributes} value_info criteria type list required 0
is_list 1 list_length non_empty}}
```

See Also

- help
- man

---

get_design

Creates a collection of one or more designs loaded into PrimeTime. You can assign these designs to a variable or pass them into another command.

Syntax

collection *get_designs*

```
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns
```

Data Types

expression	string
patterns	list

Arguments

`-hierarchical`

Searches for designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. Using this option does not force an autolink.

`-filter expression`

Filters the collection with the *expression* value. For any designs that match *patterns*, the expression is evaluated based on the design's attributes. If the expression evaluates to true, the design is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive; you can specify only one.

`patterns`

Matches design names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type design.

Description

This command creates a collection of designs from those currently loaded into PrimeTime that match certain criteria. The command returns a collection if any designs match the *patterns* option and pass the filter, if specified. If no objects matched your criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_designs* command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the *query_objects* command. In addition, you can assign the *get_designs* result to a variable.

When issued from the command prompt, the *get_designs* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable .

The "implicit query" property of the *get_designs* command provides a fast, simple way to display designs in a collection. However, if you want the flexibility provided by the

query_objects command (for example, if you want to display the object class), use the *get_designs* command as an argument to the *query_objects* command.

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

Examples

The following example queries the designs that begin with 'mpu.' Although the output looks like a list, it is just a display. A complete listing of designs is available using the *list_designs* command.

```
pt_shell> get_designs mpu*
{"mpu_0_0", "mpu_0_1", "mpu_1_0", "mpu_1_1"}
```

The following example shows that, given a collection of designs, you can remove those designs.

```
pt_shell> remove_design [get_designs mpu*]
Removing design mpu_0_0...
Removing design mpu_0_1...
Removing design mpu_1_0...
Removing design mpu_1_1...
```

See Also

- collections
- filter_collection
- list_designs
- query_objects
- remove_design
- collection_result_display_limit

---

get_design_variation

Performs design variation analysis based on Monte Carlo (MC) statistical simulation for the given path collection and generates a *design_variation* object collection.

Syntax

collection *get_design_variation*

```
path_collection
[-sample_size number]
```

Feedback

Data Types

path_collection	collection
number	int

Arguments

path_collection

Specifies the timing path collection for design variation analysis, as generated by the *get_timing_paths* command.

-sample_size *number*

Specifies the number of samples in the Monte Carlo (MC) run. The default is 10000. The range is from 5000 to 1000000000 (1 billion). Larger sample counts require more runtime and memory for simulation.

Description

This command performs design variation analysis based on Monte Carlo (MC) statistical simulation for the given path collection. The command returns a *design_variation* object. You can then use the *report_design_variation* and *report_variation_bottleneck* commands to generate design variation-related reports on the *design_variation* object.

To generate a path collection for design variation analysis, use the *get_timing_paths* command with the *-pba_mode path* or *-pba_mode exhaustive* option, and use the *-pocv_pruning* option to save runtime. The *-pocv_pruning* option prunes away subcritical paths not contributing to a high sigma failure rate.

You can specify the number of samples used in Monte Carlo simulation. The default is 10,000 samples, resulting in a *get_design_variation* runtime of about 10 minutes or less for a collection of 1 million paths. Using more samples or a larger path collection results in more runtime and memory usage. To reduce the turnaround time, it is recommended to specify 16 cores in the *set_host_options* command.

Design variation analysis requires that the *ps_enable_analysis* variable be set to *true*.

Examples

The following example shows a typical usage of the *get_design_variation* command.

```
prompt> set mypaths \\
          [get_timing_paths \\
            -pba_mode exhaustive \\
            -path_type full_clock_expanded -nworst 10 \\
            -max_paths 2000000 -pocv_pruning -slack_lesser_than 1.0]

prompt> set mydvar [get_design_variation -sample_size 100000 $mydvar]

prompt> report_design_variation -num_sigma 3.0 -nworst 10 $mydvar \\
           -significant_digits 5 -nosplit
```

Feedback

See Also

- ps_enable_analysis
- get_timing_paths
- report_design_variation
- report_variation_bottleneck
- report_cell_robustness
- report_voltage_robustness

---

get_designs

Creates a collection of one or more designs loaded into PrimeTime. You can assign these designs to a variable or pass them into another command.

Syntax

collection *get_designs*

```
[-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns
```

Data Types

expression	string
patterns	list

Arguments

`-hierarchical`

Searches for designs inferred by the design hierarchy relative to the current instance. The full name of the object at a particular level must match the patterns. Using this option does not force an autolink.

`-filter expression`

Filters the collection with the *expression* value. For any designs that match *patterns*, the expression is evaluated based on the design's attributes. If the expression evaluates to true, the design is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive; you can specify only one.

`patterns`

Matches design names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type design.

Description

This command creates a collection of designs from those currently loaded into PrimeTime that match certain criteria. The command returns a collection if any designs match the *patterns* option and pass the filter, if specified. If no objects matched your criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_designs* command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the *query_objects* command. In addition, you can assign the *get_designs* result to a variable.

When issued from the command prompt, the *get_designs* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable .

Feedback

The "implicit query" property of the *get_designs* command provides a fast, simple way to display designs in a collection. However, if you want the flexibility provided by the *query_objects* command (for example, if you want to display the object class), use the *get_designs* command as an argument to the *query_objects* command.

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

Examples

The following example queries the designs that begin with 'mpu.' Although the output looks like a list, it is just a display. A complete listing of designs is available using the *list_designs* command.

```
pt_shell> get_designs mpu*
{"mpu_0_0", "mpu_0_1", "mpu_1_0", "mpu_1_1"}
```

The following example shows that, given a collection of designs, you can remove those designs.

```
pt_shell> remove_design [get_designs mpu*]
Removing design mpu_0_0...
Removing design mpu_0_1...
Removing design mpu_1_0...
Removing design mpu_1_1...
```

See Also

- collections
- filter_collection
- list_designs
- query_objects
- remove_design
- collection_result_display_limit

---

get_distributed_variables

From the DMSA manager, gets the values of Tcl variables from workers and creates either an array of values, indexed by the scenario name, or a list of merged values.

Syntax

status *get_distributed_variables*

```
[-pre_commands pre_command_string]
[-post_commands post_command_string]
```

Feedback

```
[-attributes attribute_list]
[-merge_type min | max | average | sum | list | unique_list | none]
[-null_merge_method ignore | error | override]
variable_list
```

Data Types

```
pre_command_string          string
post_command_string         string
attribute_list              list
variable_list               list
```

Arguments

`-pre_commands pre_command_string`

Specifies a string of commands to be executed in the worker context before the retrieval of variables. Group commands with the semicolon character (;). The maximum length of the command string is 1,000 characters.

`-post_commands post_command_string`

Specifies a string of commands to be executed in the worker context after the retrieval of variables. Group commands with the semicolon character (;). The maximum length of the command string is 1,000 characters.

`-attributes attribute_list`

Specifies a list of attributes to be retrieved from a worker collection. If you do not use this option, only the *full_name*, *scenario_name*, and *object_class* attributes are retrieved. Use this option together with the *set_distributed_parameter -collection_levels* command to control the number of collection levels to traverse when retrieving attributes from each worker.

`-merge_type min | max | average | sum | list | unique_list | none`

Merges variable values that come from the scenarios as they are retrieved, producing a single value or list. You can specify any one of the following arguments:

- *min* - Gets the minimum variable value.
- *max* - Gets the maximum variable value.
- *average* - Gets the average of all variable values.
- *sum* - Gets the sum of all variable values.
- *list* - Gets a single list of all variable values. If the inputs are list variables, then a combined list is created. If the inputs are arrays, then merging is performed element by element and a single array is created. In the result, the members of the list can be in any order.

- *unique_list* - Same as the *list* option except that duplicate elements are removed from the list or array created.
- *none* (default)- Brings back the scenario variable values as an array, indexed by the scenario name.

Scalar variables, lists, and arrays of one or more dimensions can be merged, but not collection objects.

`-null_merge_method ignore | error | override`

Specifies the tool behavior when a null (empty) value of {} is encountered during the merging operations. You can specify one of the following arguments:

- *ignore* (default) - Ignores the null value.
- *error* - Issues an error message if a null value is found.
- *override* - Allows a null value to override other values.

`variable_list`

Specifies a list of variables to retrieve from the workers.

Description

The *get_distributed_variables* command retrieves variable values from the workers and makes them available for further processing at the manager. The variables specified in the *variable_list* option must be worker variables. Each worker variable can have a different value depending on the scenario context. This command can be used only in the manager process of a distributed multi-scenario analysis (DMSA) run.

By default, the command creates an array at the manager for each variable. The array has the same name as the variable and is indexed by the scenario name. Each value in the array for a given scenario index has the same value as the worker variable in that scenario's context. The variable can be a Tcl array, Tcl list, or collection.

By default, the *get_distributed_variables* command brings back a scenario variable as an array. By using the *-merge_type* option, you can merge the values back into a variable during retrieval. Merging can be especially useful when you are writing customized DMSA scripts.

When the number of scenarios is greater than the number of processors, variables can be destroyed during the save and restore process. In that situation, use the *-pre_commands* option to generate the variables to be transmitted. When retrieving a collection, use the *-attributes* option to specify what attributes are to be retrieved from the workers for the given collection.

Examples

In the following example, the number of scenarios is less than the number of processors.

```
pt_shell> remote_execute {set my_paths [get_timing_paths -nworst 10]}
pt_shell> get_distributed_variables my_paths -attributes {slack}
pt_shell> foreach_in_collection path $my_paths(scen1) {
             echo [get_attribute $path slack]
             }
pt_shell> remote_execute {set pslack [get_attribute get_pins U1/A] slack}
pt_shell> get_distributed_variables pslack
pt_shell> echo [array get pslack]
{scen1 0.1231 scen2 0.1232}
```

In the following example, the number of scenarios is greater than the number of processors.

```
pt_shell> get_distributed_variables my_paths \\
  -pre_commands {set my_paths [get_timing_paths -nworst 10]} -attributes
 slack
```

The following example shows how to access an attribute that is a collection.

```
pt_shell> remote_execute {set my_pins [get_pins *]}
pt_shell> get_distributed_variable my_pins -attributes {direction clocks}
pt_shell> foreach_in_collection pin $my_pins(scen1) {
               echo "direction:[get_attribute $pin direction]"
               set my_clocks [get_attribute $pin clocks]
               foreach_in_collection my_clock $my_clocks {
               echo "full clock name : get_attribute $my_clock
 full_name" }
          }
```

In the following example, the slack variable is set at each scenario, and the *get_distributed_variables* command returns the minimum value of the variable.

```
scenario best_case:
set slack 0.3084

scenario typical_case:
set slack 0.0901

scenario worst_case:
set slack -0.7081


pt_shell> get_distributed_variables {slack} -merge_type min
1
pt_shell> echo $slack
-0.7081
```

The following example generates a list of all clocks at all scenarios:

```
pt_shell> remote_execute {set clock_names \\
             [get_object_name [all_clocks]]}
pt_shell> get_distributed_variables clock_names -merge_type list
```

```
1
pt_shell> echo $clock_names
CLK33 CLK66 CLK50 CLK100 ATPGCLOCK JTAGCLK
```

The following example merges the my_cells variable from three scenarios with the *unique_list* mode:

```
scenario best_case:
set cells {{U1 U2} {U4 U5} {U7 U8}}
scenario type_case:
set cells {{U2 U3} {U6 U7}}
scenario worst_case:
set cells {{U1} {U5 U6}}

pt_shell> get_distributed_variables my_cells -merge_type unique_list
1
pt_shell> echo $my_cells
U4 U7 U3 U1 U5 U8 U6 U2
```

See Also

- set_distributed_parameters
- set_distributed_variables

---

get_drc_error_data

Creates a collection of DRC error data objects.

Syntax

```
collection get_drc_error_data
[-all]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect exact_count]
[-expect_at_least minimum_count]
[-expect_each_pattern_matches]
[-of_objects objects]
[patterns]
```

Data Types

```
expression          string
exact_count         int
minimum_count       int
objects             list
patterns            list
```

Feedback

Arguments

`-all`

Includes unopened DRC error data objects, if matched. The default is to return only DRC error data objects that are currently open.

`-filter` *`expression`*

Filters the collection with *expression*.

For any DRC error data objects that match the *patterns* or *objects* argument, the expression is evaluated based on the DRC error data object's attributes. If the expression evaluates to true, the DRC error data object is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

Regular expression matching is the same as in the Tcl *regexp* command. When using *-regexp*, take care in the way you quote the *patterns* argument and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

Makes matches case-insensitive.

`-exact`

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters.

The *-regexp* and *-exact* options are mutually exclusive.

`-expect` *`exact_count`*

Specifies an expected number of objects to find.

If the command finds a different number of objects, it raises a Tcl error and stops processing.

Feedback

`-expect_at_least` *`minimum_count`*

Specifies a minimum number of objects to find.

If the command finds fewer objects, it raises a Tcl error and stops processing.

`-expect_each_pattern_matches`

Specifies that each pattern in *patterns* must match at least one object.

If one or more patterns does not match, the command raises a Tcl error and stops processing.

`-of_objects` *`objects`*

Creates a collection of DRC error data objects connected to the specified objects. You can specify the following types of objects: DRC error types (drc_error_type), DRC errors (drc_error), and blocks.

The *-of_objects* option and *patterns* argument are mutually exclusive; you can specify only one. If you do not specify either, the wildcard "*" is used as the *pattern*.

`patterns`

Matches DRC error data object names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of DRC error data objects.

The *-of_objects* option and *patterns* argument are mutually exclusive; you can specify only one. If you do not specify either, the wildcard "*" is used as the *pattern*.

Description

The *get_drc_error_data* command creates a collection of DRC error data objects that match certain criteria. The command returns a collection if any DRC error data objects match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the command returns the empty string.

To include DRC error data objects that are not open, use the *-all* option. The *get_drc_error_data* command creates a collection that contains the DRC error data object, but does not open the DRC error data object. To query or modify the error type objects and error objects in the DRC error data object, you must open the DRC error data object with the *open_drc_error_data* command.

You can use the *get_drc_error_data* command at the command prompt or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *get_drc_error_data* result to a variable.

When issued from the command prompt, *get_drc_error_data* behaves as though *query_objects* had been called to display the objects in the collection. By default, a

Feedback

maximum of 100 objects is displayed; you can change this maximum by using the *shell.common.collection_result_display_limit* application option.

The "implicit query" property of the *get_drc_error_data* command provides a fast, simple way to display the DRC error data objects in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use the *get_drc_error_data* command as an argument to the *query_objects* command.

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

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example creates a collection that contains the DRC error data object named "dppinassgn.err", and then opens it.

```
prompt> set data [get_drc_error_data -all dppinassgn.err]
{"dppinassgn.err"}
prompt> open_drc_error_data $data
{"dppinassgn.err"}
```

See Also

- close_drc_error_data
- create_drc_error_data
- open_drc_error_data
- remove_drc_error_data
- save_drc_error_data

---

get_drc_error_types

Creates a collection of physical DRC error types from the given error data. You can assign these physical DRC error types to a variable or pass them into another command.

Syntax

```
collection get_drc_error_types
-error_data drc_error_data
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
```

Feedback

```
[-exact]
[-expect exact_count]
[-expect_at_least minimum_count]
[-expect_each_pattern_matches]
[patterns]
[-of_objects objects]
```

Data Types

```
drc_error_data  collection
expression      string
exact_count     int
minimum_count   int
patterns        list
objects         list
```

Arguments

`-error_data drc_error_data`

Specifies the error data in which to find objects.

`-filter expression`

Filters the collection with *expression*. For any physical DRC error types that match *patterns* (or *objects*) the expression is evaluated based on the physical DRC error type's attributes. If the expression evaluates to true, the physical DRC error type is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as a true regular expression rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with true regular expressions rather than simple wildcard patterns. *-regexp* and *-exact* are mutually exclusive.

`-nocase`

Makes matches case-insensitive.

`-exact`

Disables simple pattern matching. Use this option when searching for objects that contain the * and ? wildcard characters. *-exact* and *-regexp* are mutually exclusive.

`-expect` *`exact_count`*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

`-expect_at_least` *`minimum_count`*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

`-expect_each_pattern_matches`

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

`-of_objects` *`objects`*

Creates a collection of physical DRC error types connected to the specified objects. In this case, each object is either a named drc_error or a drc_error collection. *-of_objects* and *patterns* are mutually exclusive; you must specify one, but not both.

`patterns`

Matches physical DRC error type names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of drc_error_type. *patterns* and *-of_objects* are mutually exclusive; you must specify one, but not both. If both the *patterns* and *-of_objects* are omitted, then the wildcard "*" is used as the *pattern*

Description

The *get_drc_error_types* command creates a collection of physical DRC error types in the given error data that match certain criteria. The command returns a collection if any physical DRC errors match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl *regexp* command. When using *-regexp*, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_drc_error_types* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *get_drc_error_types* result to a variable.

Feedback

When issued from the command prompt, *get_drc_error_types* behaves as though *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the *shell.commmon.collection_result_display_limit* application option.

The "implicit query" property of *get_drc_error_types* provides a fast, simple way to display physical DRC error types in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use *get_drc_error_types* as an argument to *query_objects*.

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

Examples

The following example queries the physical DRC error type that is named "route short". Although the output looks like a list, it is not. The output is just a display. The quotation marks around (route short) are required because the type name contains a space.

```
prompt> set data [open_drc_error_data -file_name my_design.err]
prompt> get_drc_error_types -error_data $data {"route short"}
{"route short"}
```

The following example shows that, given a collection of physical DRC errors, you can query the physical DRC error types connected to those errors.

```
prompt> set error [get_drc_errors -error_data $data 9]
{"9"}
prompt> query_objects [get_drc_error_types -error_data $data -of_objects
 $error]
{"route short"}
```

The following example gets the physical DRC errors associated with an error type.

```
prompt> get_drc_errors -error_data $data -of_objects \\
        [get_drc_error_types -error_data $data {"route short"}]
{"9", "10", "11"}
1
```

See Also

- collections
- filter_collection
- query_objects

Feedback

get_drc_errors

Creates a collection of physical DRC errors from the given error data. You can assign these physical DRC errors to a variable or pass them into another command.

Syntax

```
collection get_drc_errors
-error_data drc_error_data
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-expect exact_count]
[-expect_at_least minimum_count]
[-expect_each_pattern_matches]
[patterns]
[-of_objects objects]
[-boundary region]
```

Data Types

drc_error_data	collection
expression	string
exact_count	int
minimum_count	int
objects	list
region	region
patterns	list

Arguments

`-error_data` *drc_error_data*

Specifies the error data for finding objects.

`-filter` *expression*

Filters the collection with *expression*. For any physical DRC errors that match *patterns* (or *objects*) the expression is evaluated based on the physical DRC error's attributes. If the expression evaluates to true, the physical DRC error is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators

to compare with real regular expressions rather than simple wildcard patterns. *-regexp* and *-exact* are mutually exclusive.

`-nocase`

Makes matches case-insensitive.

`-exact`

Disables simple pattern matching. For use when searching for objects that contain the * and ? wildcard characters. *-exact* and *-regexp* are mutually exclusive.

`-expect` *`exact_count`*

Specifies an expected number of objects to find. If the command finds a different number of objects, a Tcl error will be raised and command processing will stop.

`-expect_at_least` *`minimum_count`*

Specifies a minimum number of objects to find. If the command finds fewer objects, a Tcl error will be raised and command processing will stop.

`-expect_each_pattern_matches`

If this is specified, each pattern in *patterns* must match at least one object. If one or more patterns does not match, a Tcl error will be raised and command processing will stop.

`-of_objects` *`objects`*

Creates a collection of physical DRC errors connected to the specified objects. In this case, each object is either a named drc_error_type, net, pin, port, cell, pin guide, voltage area, or a collection of these objects. *-of_objects* and *patterns* are mutually exclusive; you may specify one, but not both.

`-boundary` *`region`*

Creates a collection of physical DRC errors with shapes within or touching the specified boundary shape. The region may be entered as a bounding box or as a list of coordinates. *-boundary* and *patterns* are mutually exclusive; you may specify one, but not both.

`patterns`

Matches physical DRC error names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type drc_error. *patterns* and *-of_objects* are mutually exclusive; you may specify one, but not both. *patterns* is also mutually exclusive with *-of_objects*; you may specify one, but not both. If both the *patterns* and *-of_objects* are ommitted, then the wildcard "*" is used as the *pattern*

Feedback

Description

The *get_drc_errors* command creates a collection of physical DRC errors in the given error data that match certain criteria. The command returns a collection if any physical DRC errors match the *patterns* or *objects* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the Tcl *regexp* command. When using *-regexp*, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_drc_errors* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *get_drc_errors* result to a variable.

When issued from the command prompt, *get_drc_errors* behaves as though *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum by setting the *shell.common.collection_result_display_limit* application option.

The "implicit query" property of *get_drc_errors* provides a fast, simple way to display physical DRC errors in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use *get_drc_errors* as an argument to *query_objects*.

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

Examples

The following example queries the physical DRC errors that is associated with an error type named "route short". Although the output looks like a list, it is not. The output is just a display. Note the required quotes because the type name contains a space.

```
prompt> set data [open_drc_error_data -file_name my_design.err]
prompt> get_drc_errors -error_data $data -filter {type_name == "route
 short"}
{"9", "10", "11"}
```

The following example shows that, given a collection of error types, you can query the physical DRC errors connected to those error types.

```
prompt> set type [get_drc_error_types -error_data $data {{route short}}]
{"route short"}
prompt> query_objects [get_drc_errors -error_data $data -of_objects
 $type]
{"9", "10", "11"}
```

Feedback

The following example gets the *objects* attribute for physical DRC error 9, which in this case are two nets associated with the error.

```
prompt> get_attribute [get_drc_errors -error_data $data 9] objects
{"F_12_", "F_11_"}
1
```

See Also

- collections
- filter_collection
- query_objects

---

get_em_max_capacitance

Reports electromigration (EM) maximum capacitance data from the library for a specified instance pin or library pin.

Syntax

status *get_em_max_capacitance*

```
[-related_pin pin]
[-transition_time transition_time]
[-toggle_rate toggle_rate]
[-current_type type]
object_name
```

Data Types

transition_time float *toggle_rate* float *load* float *object_name* string

Arguments

`-related_pin pin`

Reports max EM toggle rate for the specified arc.

`-transition_time float`

Specifies the transition time at the input.

`-toggle_rate float`

Specifies the toggle rate at the output pin.

`-current_type type`

Specifies the table to use for getting the maximum capacitance values. Valid values are *ave*, *rms*, and *peak*. If this option is not used, the command checks all three current types and gets the worst one.

Feedback

Description

This command reports the EM maximum capacitance from the library for a specific cell pin.

Examples

The following command reports the maximum capacitance for the pin 'U1/Z'.

```
pt_shell> get_em_max_capacitance U1/Z
```

The following command reports the maximum capacitance for the pin 'U2/Z' with the rms current type.

```
pt_shell> get_em_max_capacitance  -current_type rms U2/Z
```

See Also

- report_cell_em_violation
- get_em_max_toggle_rate

---

get_em_max_toggle_rate

Reports electromigration (EM) maximum toggle rate data from the library for a specified instance pin or library cell pin.

Syntax

status *get_em_max_toggle_rate*

```
[-related_pin pin]
[-transition_time transition_time]
[-transition_type transition_type]
[-output_load load]
[-current_type type]
[-verbose]
object_name
```

Data Types

pin	pin
transition_time	float
transition_type	max|min
load	float
type	string
object_name	string

Feedback

Arguments

`-related_pin` *pin*

Reports max EM toggle rate for the specified arc.

`-transition_time` *float*

Specifies the transition time at the input.

`-transition_type` *max|min*

Specifies the transition type at input. Valid values are *max* and *min*.

`-output_load` *float*

Specifies the load at the output pin.

`-current_type` *type*

Specifies the table to use for toggle rate calculations. Valid values are *average*, *rms*, and *peak*. If this option is not used, the command checks only the RMS table.

`-verbose`

Shows the slew rate and capacitance values used for max toggle rate violation calculations; also reports errors for missing EM data for pins.

object_name

Specifies an instance pin from the current design or a library cell pin.

Description

This command reports the EM maximum toggle rate from the library for a specific cell pin or library cell pin.

Examples

The following command reports the max toggle rate for the pin 'cell_em/test_em_cell_2/Z' with a load of 5.4.

```
pt_shell> get_em_max_toggle_rate -output_load 5.4
 cell_em/test_em_cell_2/Z
```

The following command reports the max toggle rate for the pin 'cell_em/test_em_cell_2/A' with a transition time of 0.5.

```
pt_shell> get_em_max_toggle_rate  -transition_time 0.5
 cell_em/test_em_cell_2/A
```

```
The following command reports the EM max toggle rate arc for a specified
 arc.
```

Feedback

```
pt_shell> get_em_max_toggle_rate -transition_time 2.053 -output_load
 2.053  \\
  -related_pin cell_em/test_em_cell_3/A cell_em/test_em_cell_3/Z
```

The following command reports the max toggle for the specified pin from the average type table.

```
pt_shell> get_em_max_toggle_rate -current_type ave cell_em/test_em_cell/Z
```

See Also

- report_cell_em_violation
- get_switching_activity

---

get_generated_clock

Creates a collection of generated clocks.

Syntax

collection *get_generated_clocks*

```
[-quiet]
[-regexp]
[-nocase]
[-filter expression]
patterns
```

Data Types

expression	string
patterns	list

Arguments

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-filter` *`expression`*

Filters the collection with the *expression* option. For any generated clocks that match the *patterns* option, the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

`patterns`

Matches generated clock names against patterns. Patterns can include the wildcard characters "*" and "?".

Description

The *get_generated_clocks* command creates a collection of generated clocks from the current design that match certain criteria. The command returns a collection if any generated clocks match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

To create a generated clock in the design, use the *create_generated_clock* command. To remove a generated clock from the design, use the *remove_generated_clock* command. To show information about clocks and generated clocks in the design, use the *report_clock* command.

You can use the *get_generated_clocks* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_generated_clocks* command result to a variable.

When issued from the command prompt, the *get_generated_clocks* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_generated_clocks* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command, use the *get_generated_clocks* command as an argument to the *query_objects* command. For example, use this to display the object class.

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

Examples

The following command applies the *set_clock_latency* command on all generated clocks in the design matching the "GEN*".

```
pt_shell> set_clock_latency 2.0 [get_generated_clocks "GEN*"]
```

The following command removes all the generated clocks in the design matching "GEN1*".

Feedback

```
pt_shell> remove_generated_clock [get_generated_clocks "GEN1*"]
```

See Also

- collections
- create_generated_clock
- query_objects
- remove_generated_clock
- report_clock
- collection_result_display_limit

---

get_generated_clocks

Creates a collection of generated clocks.

Syntax

collection *get_generated_clocks*

```
[-quiet]
[-regexp]
[-nocase]
[-filter expression]
patterns
```

Data Types

expression	string
patterns	list

Arguments

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-filter` *`expression`*

Filters the collection with the *expression* option. For any generated clocks that match the *patterns* option, the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

`patterns`

Matches generated clock names against patterns. Patterns can include the wildcard characters "*" and "?".

Description

The *get_generated_clocks* command creates a collection of generated clocks from the current design that match certain criteria. The command returns a collection if any generated clocks match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

To create a generated clock in the design, use the *create_generated_clock* command. To remove a generated clock from the design, use the *remove_generated_clock* command. To show information about clocks and generated clocks in the design, use the *report_clock* command.

You can use the *get_generated_clocks* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_generated_clocks* command result to a variable.

When issued from the command prompt, the *get_generated_clocks* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_generated_clocks* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command, use the *get_generated_clocks* command as an argument to the *query_objects* command. For example, use this to display the object class.

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

Examples

The following command applies the *set_clock_latency* command on all generated clocks in the design matching the "GEN*".

```
pt_shell> set_clock_latency 2.0 [get_generated_clocks "GEN*"]
```

The following command removes all the generated clocks in the design matching "GEN1*".

```
pt_shell> remove_generated_clock [get_generated_clocks "GEN1*"]
```

See Also

- collections
- create_generated_clock
- query_objects
- remove_generated_clock
- report_clock
- collection_result_display_limit

---

get_gpd_corners

Returns a string containing a space-separated list of the operating condition corners defined in a specified GPD directory.

Syntax

string *get_gpd_corners*

`-gpd` *gpd_directory*

Data Types

gpd_directory	string

Arguments

`-gpd` *gpd_directory*

Specifies the absolute or relative path to the GPD directory.

Description

This command reports the operating condition corners stored in a Galaxy Parasitics Database (GPD) directory. You can query the database without reading it into the PrimeTime tool.

Examples

The following command reports the operating corners of the GPD directory named mypara.gpd.

```
pt_shell> get_gpd_corners -gpd /mydata/mydesign/mypara.gpd
CMINW125 CMINB40 CMINN25 CMAXW125 CMAXB40 CMAXN25
```

Feedback

See Also

- get_coupling_capacitors
- get_gpd_layers
- get_ground_capacitors
- get_resistors
- report_gpd_properties

---

get_gpd_layers

Returns a string containing a space-separated list of the layer names defined in a specified GPD directory.

Syntax

string *get_gpd_layers*

```
-gpd gpd_directory
[-type routing | via | all]
[-dblayer_name]
[-layer_name]
```

Data Types

```
gpd_directory       string
```

Arguments

`-gpd gpd_directory`

Specifies the absolute or relative path to the GPD directory.

`-type routing | via | all`

Specifies the type of layers to report: *routing* layers only, *via* layers only, or *all* layers. The default is *all*.

`-dblayer_name`

Outputs layers in terms of design layer name.

`-layer_name`

Outputs layers in terms of ITF mask name.

Description

This command reports the names of the layers stored in a Galaxy Parasitics Database (GPD) directory. You can query the database without reading it into the PrimeTime tool.

Feedback

Examples

The following command reports all layers of the GPD directory named mypara.gpd.

```
pt_shell> get_gpd_layers -gpd /mydata/mydesign/mypara.gpd
SUBSTRATE poly M1 M2 M3 M4 M5 polyCont V1 V2 V3 V4
```

The following command reports the routing layers of the GPD directory named mypara.gpd.

```
pt_shell> get_gpd_layers -type routing -gpd /mydata/mydesign/mypara.gpd
SUBSTRATE poly M1 M2 M3 M4 M5
```

See Also

- get_coupling_capacitors
- get_gpd_corners
- get_ground_capacitors
- get_resistors
- report_gpd_properties

---

get_ground_capacitors

Creates a collection of parasitic ground capacitors from the specified net. You can assign these ground capacitors to a variable or pass them into another command.

Syntax

collection *get_ground_capacitors*

```
[-filter expression]
[-quiet]
[-parasitic_corners corner_list]
[-all_parasitic_corners]
[-of_objects objects]
[-from node_name]
[-to node_name]
[-unit]
[-info]
```

Data Types

expression	string
corner_list	list
objects	list
node_name	string

Feedback

Arguments

`-filter` *`expression`*

Filters the collection with the *expression* value. For any ground capacitors that match *patterns* or *objects*, the expression is evaluated based on the ground capacitor's attributes. If the expression evaluates to true, the ground capacitor is included in the result.

`-quiet`

Suppresses warning and error messages if there are no objects that match. Syntax error messages are not suppressed.

`-parasitic_corners` *`corner_list`*

Indicates the parasitic corner ordering for ground capacitance values. This option is only applicable for parasitics information obtained using *read_parasitics* with the GPD parasitic format. If the *-parasitic_corners* option is not specified, then the single parasitic corner specified by *set parasitic_corner_name* is returned.

`-all_parasitic_corners`

Specifies that all parasitic corners are to be returned. The parasitics ordering is determined according to the corner definitions of the GPD parasitics.

`-of_objects` *`objects`*

Creates a collection of ground capacitors associated with the specified objects. Each object must be a named net or net collection.

You must use either the *-of_objects* option or the *-from* and *-to* options.

`-from` *`node_name`*

Constrains the selected ground capacitors to be part of a path starting from the specified node. The node name can be either a pin, port, or an internal node, where the internal node name syntax is *net_name*:*node_id*, where the legal *node_id* range is from 1 to the number of nodes in the net. This option is used only with the *-to* option.

`-to` *`node_name`*

Constrains the selected ground capacitors to be part of a path ending with the specified node, using the same node name syntax as the *-from* option.

`-unit`

Show unit of ground_capacitors

`-info`

Get unit info of ground_capacitors

Feedback

Description

This command creates a collection of ground capacitors for the specified nets that match certain criteria. You must use either the *-of_objects* option and specify a net collection or the *-from* and *-to* options to specify a path from which to get the ground capacitors. If no objects match the criteria, an empty string is returned.

When issued from the command prompt, the *get_ground_capacitors* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

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

Examples

The following example queries the parasitic ground capacitors of net n22, subject to those objects whose max corner capacitance is greater than 0.5e-3.

```
pt_shell> get_ground_capacitors -of_objects n22 \\
  -filter "capacitance_max > 0.5e-3"
{"ground_capacitor"}
```

See Also

- collections
- filter_collection
- get_coupling_capacitors
- get_resistors
- link_design
- query_objects
- collection_result_display_limit

---

get_gui_stroke_bindings

Queries the data for stroke bindings.

Syntax

get_gui_stroke_bindings

```
[-dictionary dict_name]
[-builtin]
```

Feedback

Arguments

`-dictionary` *`dictionary_name`*

Report only the bindings for the specified dictionary.

`-builtin`

Return information on the built-in commands that are available for use in bindings.

Description

This command queries the current state of stroke bindings for the application. If no options are specified, the application returns data for all dictionaries. The *-dictionary* option limits the data returned to only that for a given dictionary. The *-builtin* option queries for the built-in commands that are available for use with bindings.

A user does not typically use this command. The tool provides built-in reporting commands that produce human-readable reports from the information returned by this command. You can use the *report_gui_stroke_bindings* and *report_gui_stroke_builtins* commands for report generation.

This command returns its data as a Tcl list, suitable for further processing into reports or command streams by Tcl procedures.

DATA FORMAT

The command returns data differently depending on the options that are provided.

The data for a dictionary is a list of entries where each entry contains the stroke sequence and the data for the command invoked by that sequence. If the command queries more than one dictionary, another level of lists is added with dictionary name followed by dictionary data.

Built-in command data is formatted as a list of commands. Each command has the name of the built-in command followed by the data for the command.

Examples

The following example returns all binding information for all dictionaries.

```
psyn_gui-t> get_gui_stroke_bindings
```

The following example returns the bindings for the Graphics dictionary.

```
psyn_gui-t> get_gui_stroke_bindings -dictionary Graphics
```

The following example returns the built-in commands available for use in bindings.

```
psyn_gui-t> get_gui_stroke_bindings -builtin
```

Feedback

See Also

- report_gui_stroke_bindings
- report_gui_stroke_builtins
- set_gui_stroke_binding
- set_gui_stroke_preferences

---

get_hier_clocks

Creates a collection of clocks captured in binary block models or top-level context data written in the HyperScale flow.

Syntax

string *get_hier_clocks*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-of_objects objects]
[patterns]
```

Data Types

expression	string
objects	list
patterns	list

Arguments

`-filter expression`

Filters the collection with *expression*. For any *hier_clock* objects that match *patterns*, the expression is evaluated based on the *hier_clock*'s attributes. If the expression evaluates to *true*, the *hier_clock* is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

`-nocase`

Makes matches case-insensitive.

patterns

Matches hier clocks names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the *wildcards* man page.

The *patterns* and *-of_objects* options are mutually exclusive.

-of_objects *cells*

Specifies a list of objects to obtain associated hierarchical clock objects from.

In a top-level analysis, this can be a list of hierarchical subblock instances from which to query how top-level clocks propagate into those blocks.

In a block-level analysis, this can be a list of clock input ports from which to query top-level clock information from the currently applied block context data.

The *patterns* and *-of_objects* options are mutually exclusive.

Description

The *get_hier_clocks* command creates a collection of clocks that are captured in the binary block model or top context data written for HyperScale flow. The command returns a collection if any clocks match the *patterns* argument and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Unlike the *get_clocks* command, this command returns *hier_clock* objects instead of *clock* objects. *hier_clock* objects are similar in that they contain attributes with clock information (such as name, period, and source), but they cannot be used in place of *clock* objects in commands that support *clock* objects.

hier_clock objects are not user-defined by *create_clock* or *create_generated_clock* commands in the constraints for the current session; but rather, they are automatically loaded as part of the binary subblock model or top-level context, which represents the clocks originally captured from user-defined *create_clock* or *create_generated_clock* commands when the block model or top-level context was generated. These *hier_clock* objects are meant to provide visibility into what clocks are expected in the binary model and context; or what clocks were defined by the model or context data provider.

You can use the *report_attribute -application* command to list all attributes defined on *hier_clock* objects.

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

Examples

The following example queries the *hier_clock* objects and creates a collection of them.

```
pt_shell> get_hier_clocks SUB*
{"SUB_CLK1", "SUB_CLK2"}
```

See Also

- set_clock_map

---

get_ilm_objects

Returns a collection of nets, cells, or pins that are part of the interface logic for the current design.

Syntax

collection *get_ilm_objects*

```
[-type net | pin | cell]
```

Arguments

`-type net | pin | cell`

Specifies the type of object to be returned as a collection of objects. Allowed values are *net*, *pin*, or *cell*. A collection is returned of objects of the specified type that belong to the interface logic on the current design.

Description

Returns a collection of nets, cells, or pins that have been identified as belonging to interface logic by using the *identify_interface_logic* command. The *is_interface_logic_pin* attribute identifies pins that are part of the interface logic, The command takes into account that the design corresponding to the interface logic model (ILM) is flattened, hierarchical nets are reduced to a single flattened net, and hierarchical pins and cells on the original design are not contained in the ILM. The objects returned are those that will be referred to in the interface logic netlist, script, and back-annotation files written using the *write_ilm_** commands. Use the *get_ilm_objects* command to review the objects that have been identified as belonging to the interface logic on the current design.

For a discussion of ILM creation and the associated commands, see the *identify_interface_logic* man page.

Examples

The following example returns a collection of all interface logic cells.

```
pt_shell> get_ilm_objects -type cell
```

The following command returns a collection of all nets in the flattened ILM netlist.

```
pt_shell> get_ilm_objects -type net
```

Feedback

See Also

- identify_interface_logic
- write_ilm_netlist
- write_ilm_parasitics
- write_ilm_script
- write_ilm_sdf

---

get_latch_loop_groups

Returns a list of collections of pins, each collection containing the transparent latch data pins contained within one latch loop group.

Syntax

list *get_latch_loop_groups*

```
[-of_objects pin_list]
[-loop_breakers_only]
```

Data Types

pin_list list

Arguments

`-of_objects pin_list`

Specifies a collection of data pins of transparent latches. This option returns only pins of loop groups containing the specified pins. By default, the command returns one collection for each loop group in the design.

`-loop_breakers_only`

Specifies that the returned collection contains only pins that are loop breaker latch data pins (that have a *true* value for the *is_latch_loop_breaker* pin attribute). By default, all transparent latch data pins within the loop group are returned in the collections.

Description

When sequential loops of transparent latches exist in a design, a loop group containing all latch data pins of intersecting loops is formed. The *get_latch_loop_groups* command analyzes the design and returns a collection for each loop group formed. The collection contains the latch data pins of the loop group. Combinational pins are not included in the results.

Feedback

You can use the results of the *get_latch_loop_groups* command to find paths within loops that violate timing. For an example, see the EXAMPLES section.

Before using the *get_latch_loop_groups* command, you must set the *timing_enable_through_paths* variable to *true*.

Examples

The following example reports violating paths within latch loops by using the *get_latch_loop_groups* command. No paths outside the loops are reported.

```
pt_shell> foreach a_loop_group [get_latch_loop_groups
 -loop_breakers_only] {
            report_timing -from $a_loop_group -to $a_loop_group
 -slack_lesser_than 0.0
          }
```

See Also

- report_latch_loop_groups
- timing_enable_through_paths

---

get_lib

Creates a collection of libraries loaded into PrimeTime. You can assign these libraries to a variable or pass them into another command.

Syntax

collection *get_libs*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-of_objects objects]
[patterns]
```

Data Types

expression	string
objects	list
patterns	list

Feedback

Arguments

`-filter` *`expression`*

Filters the collection with the *expression* option. For any libraries that match *patterns* (or *objects*), the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modify the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`-of_objects` *`objects`*

Creates a collection of libraries that contain the specified objects. In this case, each object is either a named library cell or a library cell collection. the *-of_objects* and *patterns* options are mutually exclusive; you can specify only one.

`patterns`

Matches library names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type lib. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

Description

The *get_libs* command creates a collection of libraries from those currently loaded into PrimeTime that match certain criteria. The command returns a collection if any libraries match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Feedback

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* option and filter the *expression* option. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_libs* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_libs* command result to a variable.

When issued from the command prompt, the *get_libs* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_libs* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command, use the *get_libs* command as an argument to the *query_objects* command. For example, use this if you want to display the object class.

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

Examples

The following example queries all loaded libraries. Although the output looks like a list, it is just a display. Note that a complete listing of libraries is available using the *list_libs* command.

```
pt_shell> get_libs *
{"misc_cmos", "misc_cmos_io"}
```

The following example shows that given a library collection, you can remove those libraries. Note that you cannot remove libraries if they are referenced by a design.

```
pt_shell> remove_lib [get_libs misc*]
Removing library misc_cmos...
Removing library misc_cmos_io...
```

See Also

- collections
- filter_collection
- list_libs
- query_objects

Feedback

- remove_lib
- collection_result_display_limit

---

get_lib_cell

Creates a collection of library cells from libraries loaded into PrimeTime. You can assign these library cells to a variable or pass them into another command.

Syntax

collection *get_lib_cells*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-of_objects objects]
patterns
```

Data Types

expression	string
objects	list
patterns	list

Arguments

`-filter expression`

Filters the collection with the *expression* value. For any library cells that match the *patterns* or *objects* argument, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`-of_objects` *objects*

Creates a collection of library cells that are referenced by the specified cells or own the specified library pins. In this case, each object is either a named library pin or netlist cell, or a library pin collection or a netlist cell collection. The *-of_objects* and *patterns* options are mutually exclusive; you must specify one, but not both.

patterns

Matches library cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type lib_cell. The *patterns* and *-of_objects* options are mutually exclusive; you must specify one, but not both.

Description

This command creates a collection of library cells from libraries currently loaded into PrimeTime that match certain criteria. The command returns a collection if any library cells match the *patterns* or *objects* option and pass the filter, if specified. If no objects match the criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_lib_cells* command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the *query_objects* command. In addition, you can assign the *get_lib_cells* command result to a variable.

When issued from the command prompt, the *get_lib_cells* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_lib_cells* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command (for example, if you want to display the object class), use the *get_lib_cells* command as an argument to the *query_objects* command.

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

Examples

The following example queries all library cells that are in the misc_cmos library and begin with AN2. Although the output looks like a list, it is just a display.

```
pt_shell> get_lib_cells misc_cmos/AN2*
{"misc_cmos/AN2", "misc_cmos/AN2P"}
```

The following example shows one way to find out the library cell used by a particular cell.

```
pt_shell> get_lib_cells -of_objects [get_cells o_reg1]
{"misc_cmos/FD2"}
```

See Also

- collections
- filter_collection
- get_cells
- query_objects
- collection_result_display_limit

---

get_lib_cells

Creates a collection of library cells from libraries loaded into PrimeTime. You can assign these library cells to a variable or pass them into another command.

Syntax

collection *get_lib_cells*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-of_objects objects]
patterns
```

Data Types

expression	string
objects	list
patterns	list

Feedback

Arguments

`-filter` *`expression`*

Filters the collection with the *expression* value. For any library cells that match the *patterns* or *objects* argument, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`-of_objects` *`objects`*

Creates a collection of library cells that are referenced by the specified cells or own the specified library pins. In this case, each object is either a named library pin or netlist cell, or a library pin collection or a netlist cell collection. The *-of_objects* and *patterns* options are mutually exclusive; you must specify one, but not both.

`patterns`

Matches library cell names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type lib_cell. The *patterns* and *-of_objects* options are mutually exclusive; you must specify one, but not both.

Description

This command creates a collection of library cells from libraries currently loaded into PrimeTime that match certain criteria. The command returns a collection if any library cells

match the *patterns* or *objects* option and pass the filter, if specified. If no objects match the criteria, an empty string is returned.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_lib_cells* command at the command prompt, or you can nest it as an argument to another command. For example, you can use it in the *query_objects* command. In addition, you can assign the *get_lib_cells* command result to a variable.

When issued from the command prompt, the *get_lib_cells* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_lib_cells* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command (for example, if you want to display the object class), use the *get_lib_cells* command as an argument to the *query_objects* command.

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

Examples

The following example queries all library cells that are in the misc_cmos library and begin with AN2. Although the output looks like a list, it is just a display.

```
pt_shell> get_lib_cells misc_cmos/AN2*
{"misc_cmos/AN2", "misc_cmos/AN2P"}
```

The following example shows one way to find out the library cell used by a particular cell.

```
pt_shell> get_lib_cells -of_objects [get_cells o_reg1]
{"misc_cmos/FD2"}
```

See Also

- collections
- filter_collection
- get_cells

Feedback

- query_objects
- collection_result_display_limit

---

get_lib_pin

Creates a collection of library cell pins from libraries loaded into PrimeTime. You can assign these library cell pins to a variable or pass them into another command.

Syntax

collection *get_lib_pins*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[patterns | -of_objects objects]
```

Data Types

expression	string
patterns	list
objects	list

Arguments

`-filter expression`

Filters the collection with the *expression* option. For any library cell pins that match the *patterns* option (or the *objects* option), the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library pin is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* option and the *-exact* option are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`patterns`

Matches library cell pin names against patterns. The *patterns* option can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. The *patterns* option can also include collections of type *lib_pin*. The *patterns* and *-of_objects* options are mutually exclusive; you must specify one, but not both.

`-of_objects` *`objects`*

Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. In this case, each object is either a named library cell or netlist pin, or a library cell collection or netlist pin collection. The *-of_objects* and *patterns* options are mutually exclusive; you must specify one, but not both.

Description

The *get_lib_pins* command creates a collection of library cell pins from libraries currently loaded into PrimeTime that match certain criteria. The command returns a collection if any library cell pins match the *patterns* or *objects* options and pass the filter (if specified). If no objects matched the criteria, the empty string is returned.

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* option and filter the *expression* option. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_lib_pins* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_lib_pins* command result to a variable.

When issued from the command prompt, the *get_lib_pins* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_lib_pins* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by

Feedback

the *query_objects* command, use the *get_lib_pins* command as an argument to the *query_objects* command. For example, use this if you want to display the object class.

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

Examples

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
pt_shell> [get_lib_pins misc_cmos/AN2/*
{"misc_cmos/AN2/A", "misc_cmos/AN2/B", "misc_cmos/AN2/Z"}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist.

```
pt_shell> get_lib_pins -of_objects o_reg1/Q
{"misc_cmos/FD2/Q"}
```

See Also

- collections
- filter_collection
- get_libs
- get_lib_cells
- query_objects
- collection_result_display_limit

---

get_lib_pins

Creates a collection of library cell pins from libraries loaded into PrimeTime. You can assign these library cell pins to a variable or pass them into another command.

Syntax

collection *get_lib_pins*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[patterns | -of_objects objects]
```

Feedback

Data Types

expression	string
patterns	list
objects	list

Arguments

`-filter` *expression*

Filters the collection with the *expression* option. For any library cell pins that match the *patterns* option (or the *objects* option), the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library pin is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* option and the *-exact* option are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

patterns

Matches library cell pin names against patterns. The *patterns* option can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. The *patterns* option can also include collections of type *lib_pin*. The *patterns* and *-of_objects* options are mutually exclusive; you must specify one, but not both.

`-of_objects` *objects*

Creates a collection of library cell pins referenced by the specified netlist pins or owned by the specified library cells. In this case, each object is either a named library cell or netlist pin, or a library cell collection or netlist pin collection. The *-of_objects* and *patterns* options are mutually exclusive; you must specify one, but not both.

Feedback

Description

The *get_lib_pins* command creates a collection of library cell pins from libraries currently loaded into PrimeTime that match certain criteria. The command returns a collection if any library cell pins match the *patterns* or *objects* options and pass the filter (if specified). If no objects matched the criteria, the empty string is returned.

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* option and filter the *expression* option. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_lib_pins* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_lib_pins* command result to a variable.

When issued from the command prompt, the *get_lib_pins* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_lib_pins* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command, use the *get_lib_pins* command as an argument to the *query_objects* command. For example, use this if you want to display the object class.

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

Examples

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
pt_shell> [get_lib_pins misc_cmos/AN2/*
{"misc_cmos/AN2/A", "misc_cmos/AN2/B", "misc_cmos/AN2/Z"}
```

The following example shows one way to find out how the library pin is used by a particular pin in the netlist.

```
pt_shell> get_lib_pins -of_objects o_reg1/Q
{"misc_cmos/FD2/Q"}
```

Feedback

See Also

- collections
- filter_collection
- get_libs
- get_lib_cells
- query_objects
- collection_result_display_limit

---

get_lib_timing_arcs

Creates a collection of library arcs for custom reporting and other processing. You can assign these library arcs to a variable and get the desired attribute for further processing.

Syntax

string *get_lib_timing_arcs*

```
[-to to_list]
[-from from_list]
[-of_objects object_list]
[-filter expression]
[-quiet]
```

Data Types

to_list	list
from_list	list
object_list	list
expression	string

Arguments

`-to` *to_list*

Specifies the "to" library pins, or ports. All backward library arcs from the specified library pins or ports are considered.

`-from` *from_list*

Specifies the "from" library pins, or ports. All forward library arcs from the specified library pins or ports are considered.

`-of_objects` *object_list*

Specifies library cells or timing arcs. If a library cell is specified, all library cell arcs of that cell are considered. If a timing arc collection is given in the object list, the corresponding library timing arc is considered.

`-filter` *`expression`*

Specifies the filter expression. A filter expression is a string that comprises a series of logical expressions describing a set of constraints you want to place on the collection of library arcs. Each subexpression of a filter expression is a relation contrasting an attribute name with a value, by means of an operator.

`-quiet`

Specifies that all messages are to be suppressed.

Description

Creates a collection of library arcs for custom reporting and other processing. You can assign these library arcs to a variable and get the desired attribute for further processing. Use the *foreach_in_collection* command to iterate among the library arcs in the collection. You can use the *get_attribute* command to obtain information about the paths. You can also use the filter expression to filter the library arcs that satisfy the specified conditions.

The following attributes are supported on library timing arcs:

```
from_lib_pin
is_disabled
is_user_disabled
mode
object_class
sdf_cond
sense
to_lib_pin
```

One attribute of a library timing arc is the *from_lib_pin*, which is the library pin from which the timing arc begins. In the same way, *to_lib_pin* is the library pin to which the timing arc ends. To get more information on *from_lib_pin* and *to_lib_pin*, use the *get_attributes* command. The *object_class* attribute holds the class name of library timing arcs: *lib_timing_arc*.

See the *collections* and *foreach_in_collection* man pages for more information.

Examples

The following procedure is an example you can use of how the collection returned from an invocation of *get_lib_timing_arcs* can be used to provide useful and readable cell level arc information.

```
proc show_lib_arcs {args} {
  set lib_arcs [eval [concat get_lib_timing_arcs $args]]
  echo [format "%15s    %-15s  %18s" "from_lib_pin" "to_lib_pin" \\
            "sense"]
  echo [format "%15s    %-15s  %18s" "------------" "----------" \\
            "-----"]

  foreach_in_collection lib_arc $lib_arcs {
```

```
    set fpin [get_attribute $lib_arc from_lib_pin]
    set tpin [get_attribute $lib_arc to_lib_pin]
    set sense [get_attribute $lib_arc sense]
    set from_lib_pin_name [get_attribute $fpin base_name]
    set to_lib_pin_name [get_attribute $tpin base_name]
    echo [format "%15s -> %-15s  %18s" \\
        $from_lib_pin_name $to_lib_pin_name \\
        $sense]
  }
}
```

```
pt_shell> show_lib_arc -of_objects [get_timing_arcs -of_objects ffa]

   from_lib_pin    to_lib_pin                    sense
   ------------    ----------                    -----
             CP -> D                    setup_clk_rise
             CP -> D                     hold_clk_rise
             CP -> Q                       rising_edge
             CP -> QN                      rising_edge
             CD -> Q                         clear_low
             CD -> QN                       preset_low
```

```
pt_shell> show_lib_arc -of_objects mylib/AN2

   from_lib_pin    to_lib_pin                    sense
   ------------    ----------                    -----
              A -> Z                    positive_unate
              B -> Z                    positive_unate
```

```
pt_shell> show_lib_arc -from mylib/AN2/A

   from_lib_pin    to_lib_pin                    sense
   ------------    ----------                    -----
              A -> Z                    positive_unate
```

See Also

- collections
- filter_collection
- foreach_in_collection
- get_attribute
- get_timing_arcs

Feedback

get_libs

Creates a collection of libraries loaded into PrimeTime. You can assign these libraries to a variable or pass them into another command.

Syntax

collection *get_libs*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-of_objects objects]
[patterns]
```

Data Types

```
expression       string
objects          list
patterns         list
```

Arguments

`-filter expression`

Filters the collection with the *expression* option. For any libraries that match *patterns* (or *objects*), the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modify the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`-of_objects` *objects*

Creates a collection of libraries that contain the specified objects. In this case, each object is either a named library cell or a library cell collection. the *-of_objects* and *patterns* options are mutually exclusive; you can specify only one.

patterns

Matches library names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type lib. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

Description

The *get_libs* command creates a collection of libraries from those currently loaded into PrimeTime that match certain criteria. The command returns a collection if any libraries match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* option and filter the *expression* option. Using rigid quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_libs* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_libs* command result to a variable.

When issued from the command prompt, the *get_libs* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_libs* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command, use the *get_libs* command as an argument to the *query_objects* command. For example, use this if you want to display the object class.

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

Feedback

Examples

The following example queries all loaded libraries. Although the output looks like a list, it is just a display. Note that a complete listing of libraries is available using the *list_libs* command.

```
pt_shell> get_libs *
{"misc_cmos", "misc_cmos_io"}
```

The following example shows that given a library collection, you can remove those libraries. Note that you cannot remove libraries if they are referenced by a design.

```
pt_shell> remove_lib [get_libs misc*]
Removing library misc_cmos...
Removing library misc_cmos_io...
```

See Also

- collections
- filter_collection
- list_libs
- query_objects
- remove_lib
- collection_result_display_limit

---

get_license

Obtains a license for a feature.

Syntax

```
status get_license
   [-quantity num_licenses]
   feature_list
```

Data Types

```
num_licenses      integer
feature_list      list
```

Arguments

`-quantity num_licenses`

Specifies the total number of licenses to check out for each feature. If a license is already checked out, the command acquires only the additional licenses

Feedback

needed to bring the total to the specified quantity. If you do not use this option, the command checks out one license per feature.

This option is useful if you need to reserve multiple licenses for a multicore run. You can reserve the required number of licenses early in the session, rather than risking a failure or a performance degradation later in the session if the licenses are not available.

feature_list

Specifies a list of features to obtain. To specify more than one feature, enclose the list enclosed in curly braces ({ }).

Description

This command obtains licenses for the specified features. These features remain checked out by you until you run the *remove_license* command or until you exit the program.

To list the features that you currently have checked out, run the *list_licenses* command.

Examples

This example obtains two licenses of the PrimePower feature:

```
pt_shell> get_license PrimePower -quantity 2
Information: Checked out license 'PrimePower' (PT-019)
Information: Checked out license 'PrimePower' (PT-019)
1
```

See Also

- license_users
- list_licenses
- remove_license

---

get_message_ids

Get application message ids

Syntax

string *get_message_ids* [-type *severity*] [*pattern*]

```
string severity
string pattern
```

Feedback

Arguments

`-type severity`

Filter ids based on type (Values: Info, Warning, Error, Severe, Fatal)

pattern

Get IDs matching pattern (default: *)

Description

The *get_message_ids* command retrieves the error, warning and informational messages used by the application. The result of this command is a Tcl formatted list of all message ids. Information about the id can be queried with the *get_message_info* command.

Examples

The following code finds all error messages and makes the application stop script execution when one of these messages is encountered.

```
foreach id [get_message_ids -type Error] {
  set_message_info -stop_on -id $id
}
```

See Also

- print_message_info
- set_message_info
- suppress_message

---

get_message_info

Returns information about tool error, warning, and information messages.

Syntax

integer *get_message_info*

```
-error_count | -warning_count | -info_count
   | -limit message_id | -save_limit message_id
   | -occurrences message_id | -suppressed message_id
   | -id message_id
```

Data Types

message_id string

Feedback

Arguments

`-error_count`

Returns the number of error messages issued so far.

`-warning_count`

Returns the number of warning messages issued so far.

`-info_count`

Returns the number of information messages issued so far.

`-limit` *`message_id`*

Returns the current user-specified limit for a given message ID, as previously set with the *set_message_info -limit* command.

`-save_limit` *`message_id`*

Returns the current *save_limit* value for a given message ID, as previously set with the *set_message_info -save_limit* command.

`-occurrences` *`message_id`*

Returns the number of occurrences of a given message ID.

`-suppressed` *`message_id`*

Returns the number of times a message was suppressed either due to a *suppress_message* specification (complete exclusion) or due to exceeding a numerical limit.

`-id` *`message_id`*

Returns information about the specified message. The information is returned as a pairwise Tcl list compatible with the *array set* command.

Description

The *get_message_info* command retrieves information about error, warning, and information messages.

For example, if the following message is issued by the tool, information about its issuance is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It can be useful to be able to retrieve recorded information about issued tool messages. For example, you can stop a script if a certain type of error has occurred, or if a more than a certain number of that error has occurred, or monitor the number of messages issued by a single command.

Feedback

You can also find out how many times a specific message has occurred, or how many times it has been suppressed, or whether an issuance or save limit has been specified for it.

This command has multiple mutually-exclusive options. One option must be specified; multiple options cannot be specified.

Examples

The following example uses the *get_message_info* command to count the number of errors that occurred during execution of a specific command, and to return from the procedure if the error count exceeds a given amount:

```
prompt> proc do_command {limit} {
    set current_errors [get_message_info -error_count]
    command
    set new_errors [get_message_info -error_count]
    if {[expr $new_errors - $current_errors] > $limit} {
      return -code error "Too many errors"
    }
    ...
  }
```

The following example uses the *get_message_info* command to retrieve information on the CMD-014 message:

```
prompt> get_message_info -id CMD-014
id CMD-014 severity Error limit 10000 save_limit -1 occurrences 0
 suppressed 0 message
{Invalid %s value '%s' in list.}
```

See Also

- print_message_info
- set_message_info
- suppress_message
- get_message_ids

---

get_message_instances

Returns a collection of recorded message instances of the specified message id.

Syntax

collection *get_message_instances*

```
[message_id]
```

Feedback

Data Types

message_id string

Arguments

message_id

Returns instances of the specified message_id as collection.

Description

This command creates a collection of saved message instances of the specified message id. This command returns a collection if the specified *message_id* has any saved instances. If no saved instances of this *message_id* exist, a empty string is returned.

The number of instances of a message to be saved has to be specified using the *save_limit* option of the *set_message_info* command.

The attributes of message instances can be queried by iterating through the objects of the message instance collection. For example, the attributes arguments, message_id and message_body can be queried on each message instance. The *arguments* attribute of the *message_instance* object class returns comma separated list of arguments of the specific message instance.

Examples

The following example uses the *get_message_instances* command to get a collection of message instances of DES-121 message, iterates through the message instances and queries attributes.

```
pt_shell> set coll  [get_message_instances DES-121]
pt_shell> foreach_in_collection itr $coll {
    echo [get_attribute -class message_instance $itr message_id]
    echo [get_attribute -class message_instance $itr arguments]
    echo [get_attribute -class message_instance $itr message_body]
}
```

See Also

- print_message_info
- set_message_info
- suppress_message
- get_message_ids

Feedback

get_net

Creates a collection of nets from the netlist. You can assign these nets to a variable or pass them into another command.

Syntax

collection *get_nets*

```
[-hierarchical]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-top_net_of_hierarchical_group]
[-segments]
[-boundary_type btype]
[-of_objects objects]
patterns
```

Data Types

expression	string
btype	string
objects	list
patterns	list

Arguments

`-hierarchical`

Searches for nets level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX *find* command. For example, if there is a net block1/muxsel, a hierarchical search would find it using "muxsel." You cannot use the *-hierarchical* option with the *-of_objects* option.

`-filter expression`

Filters the collection with *expression*. For any nets that match *patterns* (or *objects*), the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the net is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to

compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-exact* option, makes matches case-insensitive. You can use *-nocase* only when you also use the *-exact* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`-top_net_of_hierarchical_group`

Keep only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved with the collection. In the case of multiple nets at the same level, the first net specified is kept. Although this option can be used when there are multiple nets specified, it is best used in combination with the *-segments* option and with a single net.

`-segments`

Return all global segments for given nets. This modifies the initial search which matches *patterns* or *objects*. It is applied after filtering, but before the *-top_net_of_hierarchical_group* option is determined. For each net, all global segments which are part of that net will be added to the result collection. Global net segments are all those physically connected across all hierarchical boundaries. Although this option can be used with other options and when there are multiple nets specified, it is best used with a single net. When combined with the *-top_net_of_hierarchical_group* option, you can isolate the highest net segment of a physical net.

`-boundary_type` *`btype`*

Specifies what to do when getting nets of boundary pins. This option requires the *-of_objects* option. Allowed values are lower, upper, and both, meaning the net inside the hierarchical block, outside the hierarchical block, or both nets, respectively. The option has no meaning for non-hierarchical pins. Note that The "upper" value is less useful, as getting pins of a hierarchical net will return the pin outside the hierarchical block, and a subsequent get_nets would find the upper hierarchical net. Using upper on a hierarchical pin is the same as omitting the option.

`-of_objects` *`objects`*

Creates a collection of nets connected to the specified objects. Each object is either a named pin, a pin collection, a named cell or cell collection. The

-of_objects and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

patterns

Matches net names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type net. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

Description

The *get_nets* command creates a collection of nets in the current design, relative to the current instance that match certain criteria. The command returns a collection if any nets match the *patterns* or *objects* and pass the filter (if specified). If no objects matched the criteria, the empty collection is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_nets* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *get_nets* result to a variable.

When issued from the command prompt, the *get_nets* command behaves as though *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of *get_nets* provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use *get_nets* as an argument to *query_objects*.

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

Examples

The following example queries the nets that begin with 'NET' in block 'block1'. Although the output looks like a list, it is just a display.

Feedback

```
pt_shell> get_nets block1/NET
{"block1/NET1QNX", "block1/NET2QNX"}
```

The following example shows that with a collection of pins, you can query the nets connected to those pins.

```
pt_shell> current_instance block1
block1
pt_shell> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{"o_reg1/QN", "o_reg2/QN"}
pt_shell> query_objects [get_nets -of_objects $pinsel]
{"NET1QNX", "NET2QNX"}
```

This example shows how to use the *-top_net_of_hierarchical_group* and *-boundary_type* options. Given the following circuit:

```
u1                                    u2

-----------------+              +-------------------

                 |              |

 u1/Z -- net5 ---+--- topnet ---+--- net7 -- u1/A

                 |              |

-----------------+              +-------------------
```

there are hierarchical cells u1 and u2 at this level, connected by a net *topnet*. Within u1 is a pin u1/Z driving *net5*, and withing u2 is a pin u1/A being driven by *net7*. These three nets are physically the same net. Assume these are the only nets in the design. Notice the difference in the results of the following two *get_nets* commands:

```
pt_shell> get_nets * -hierarchical
{"topnet", "u1/net5", "u2/net7"}
pt_shell> get_nets * -hierarchical -top_net_of_hierarchical_group
{"topnet"}
```

Assume that at the top level, topnet is connected to pins u2/in and u1/out. Here are some examples of the *-boundary_type* option. Notice now in this case, the "upper" type does not add value.

```
pt_shell> get_nets -of_objects u2/in
{"topnet"}
pt_shell> get_nets -of_objects u2/in -boundary_type upper
{"topnet"}
pt_shell> get_nets -of_objects u2/in -boundary_type lower
{"u2/net7"}
pt_shell> get_nets -of_objects u2/in -boundary_type both
{"u2/net7", "topnet"}
pt_shell> get_nets -boundary lower -of_objects \\
```

Feedback

```
                [get_pins -of_objects [get_nets topnet]]
{"u1/net5", "u2/net7"}
```

See Also

- collections
- filter_collection
- get_pins
- link_design
- query_objects
- collection_result_display_limit

---

get_nets

Creates a collection of nets from the netlist. You can assign these nets to a variable or pass them into another command.

Syntax

collection *get_nets*

```
[-hierarchical]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-top_net_of_hierarchical_group]
[-segments]
[-boundary_type btype]
[-of_objects objects]
patterns
```

Data Types

expression	string
btype	string
objects	list
patterns	list

Arguments

`-hierarchical`

Searches for nets level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the UNIX *find* command. For example, if there is a net block1/muxsel, a

hierarchical search would find it using "muxsel." You cannot use the *-hierarchical* option with the *-of_objects* option.

`-filter` *`expression`*

Filters the collection with *expression*. For any nets that match *patterns* (or *objects*), the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the net is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-exact* option, makes matches case-insensitive. You can use *-nocase* only when you also use the *-exact* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the * and ? wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`-top_net_of_hierarchical_group`

Keep only the top net of a hierarchical group. When more than one hierarchical net of the same group is specified (local nets at various hierarchical levels of the same physical net), only the net closest to the top of the hierarchy is saved with the collection. In the case of multiple nets at the same level, the first net specified is kept. Although this option can be used when there are multiple nets specified, it is best used in combination with the *-segments* option and with a single net.

`-segments`

Return all global segments for given nets. This modifies the initial search which matches *patterns* or *objects*. It is applied after filtering, but before the *-top_net_of_hierarchical_group* option is determined. For each net, all global segments which are part of that net will be added to the result collection. Global net segments are all those physically connected across all hierarchical boundaries. Although this option can be used with other options and when there are multiple nets specified, it is best used with a single net. When combined

with the *-top_net_of_hierarchical_group* option, you can isolate the highest net segment of a physical net.

`-boundary_type` *`btype`*

Specifies what to do when getting nets of boundary pins. This option requires the *-of_objects* option. Allowed values are lower, upper, and both, meaning the net inside the hierarchical block, outside the hierarchical block, or both nets, respectively. The option has no meaning for non-hierarchical pins. Note that The "upper" value is less useful, as getting pins of a hierarchical net will return the pin outside the hierarchical block, and a subsequent get_nets would find the upper hierarchical net. Using upper on a hierarchical pin is the same as omitting the option.

`-of_objects` *`objects`*

Creates a collection of nets connected to the specified objects. Each object is either a named pin, a pin collection, a named cell or cell collection. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

`patterns`

Matches net names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type net. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

Description

The *get_nets* command creates a collection of nets in the current design, relative to the current instance that match certain criteria. The command returns a collection if any nets match the *patterns* or *objects* and pass the filter (if specified). If no objects matched the criteria, the empty collection is returned.

If any *patterns* (or *objects*) fail to match any objects and the current design is not linked, the design automatically links.

Regular expression matching is the same as in the Tcl *regexp* command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_nets* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *get_nets* result to a variable.

When issued from the command prompt, the *get_nets* command behaves as though *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of *get_nets* provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use *get_nets* as an argument to *query_objects*.

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

Examples

The following example queries the nets that begin with 'NET' in block 'block1'. Although the output looks like a list, it is just a display.

```
pt_shell> get_nets block1/NET
{"block1/NET1QNX", "block1/NET2QNX"}
```

The following example shows that with a collection of pins, you can query the nets connected to those pins.

```
pt_shell> current_instance block1
block1
pt_shell> set pinsel [get_pins {o_reg1/QN o_reg2/QN}]
{"o_reg1/QN", "o_reg2/QN"}
pt_shell> query_objects [get_nets -of_objects $pinsel]
{"NET1QNX", "NET2QNX"}
```

This example shows how to use the *-top_net_of_hierarchical_group* and *-boundary_type* options. Given the following circuit:

```
u1                                      u2

-----------------+              +-------------------

                 |              |

 u1/Z -- net5 ---+--- topnet ---+--- net7 -- u1/A

                 |              |

-----------------+              +-------------------
```

there are hierarchical cells u1 and u2 at this level, connected by a net *topnet*. Within u1 is a pin u1/Z driving *net5*, and withing u2 is a pin u1/A being driven by *net7*. These three

Feedback

nets are physically the same net. Assume these are the only nets in the design. Notice the difference in the results of the following two *get_nets* commands:

```
pt_shell> get_nets * -hierarchical
{"topnet", "u1/net5", "u2/net7"}
pt_shell> get_nets * -hierarchical -top_net_of_hierarchical_group
{"topnet"}
```

Assume that at the top level, topnet is connected to pins u2/in and u1/out. Here are some examples of the *-boundary_type* option. Notice now in this case, the "upper" type does not add value.

```
pt_shell> get_nets -of_objects u2/in
{"topnet"}
pt_shell> get_nets -of_objects u2/in -boundary_type upper
{"topnet"}
pt_shell> get_nets -of_objects u2/in -boundary_type lower
{"u2/net7"}
pt_shell> get_nets -of_objects u2/in -boundary_type both
{"u2/net7", "topnet"}
pt_shell> get_nets -boundary lower -of_objects \\
             [get_pins -of_objects [get_nets topnet]]
{"u1/net5", "u2/net7"}
```

See Also

- collections
- filter_collection
- get_pins
- link_design
- query_objects
- collection_result_display_limit

---

get_noise_violation_sources

Creates a collection of noise violation sources for custom reporting and other processing. You can assign these pins to a variable and get the desired attribute for further processing.

Syntax

integer *get_noise_violation_sources*

```
[-above]
[-below]
[-low]
[-high]
```

Feedback

```
[-nworst_endpoints pin_count]
[-max_sources_per_endpoint pin_count]
[-slack_type area | height | area_percent]
[object_list]
```

Data Types

```
pin_count       integer
object_list     list
```

Arguments

`-above`

Performs the reporting only above the rails. If this option is combined with the *-low* option, it reports the noise bumps above the low rail. If it is combined with the *-high* option, it reports the noise bumps above the high rail. Otherwise, it reports the noise bumps above the high rail and above the low rail.

`-below`

Performs the reporting only below the rails. If this option is combined with the*-low* option, it reports the noise bumps below the low rail. If this option is combined with the*-high* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps below the high rail and below the low rail.

`-low`

Performs the reporting only for the low rail. If this option is combined with the *-above* option, it reports the noise bumps above the low rail. If it is combined with the*-below* option, it reports the noise bumps below the low rail. Otherwise, it reports the noise bumps for both below and above the low rail.

`-high`

Performs the reporting only for the high rail. If this option is combined with the *-above* option, it reports the noise bumps above the high rail. If it is combined with the *-below* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps for both below and above the high rail.

`-nworst_endpoints` *`pin_count`*

Specifies the number of endpoint pins to be reported. The default is 1.

`-max_sources_per_endpoint` *`pin_count`*

Specifies the number of violation source pins per endpoint to be reported. The default is 1.

`-slack_type area | height | area_percent`

Specifies the type of slack to be used. The voltage margin is defined by the noise bump height and noise immunity curves or DC noise margin.

Feedback

- *height* (default) - Reports noise slack as the voltage margin.
- *area* - Reports noise slack as the voltage margin multiplied by the noise bump width.
- *area_percent* - Reports noise slack as the percentage of the noise constraint area. The noise constraint area is computed by multiplying the noise height constraint by the noise bump width.

object_list

Specifies the load pins for which the noise reporting is performed. If no pin is specified, reporting is performed on the entire design.

Description

Provides a collection of noise violation source pins for failing noise endpoints in the current design.

Examples

The following example creates a collection of the worst violation source for the worst endpoint in the current design and assigns it to the viol_pins variable.

```
pt_shell> set viol_pins [get_noise_violation_sources]
```

The following example creates a collection of maximum 10 violation sources for the endpoint ffa/Q above the low rail.

```
pt_shell> get_noise_violation_sources -above -low \\
           -max_sources_per_endpoint 10 [get_pin ffa/Q]
```

The following example creates a collection of maximum 10 violation source for the worst two endpoints.

```
pt_shell> get_noise_violation_sources -max_sources_per_endpoint 10 \\
           -nworst_endpoints 2
```

See Also

- report_noise
- report_noise_violation_sources
- set_noise_parameters
- update_noise

---

get_object_name

Gets the name of objects in the given collection.

Feedback

Syntax

string *get_object_name*

```
        collection
```

Data Types

```
collection       string
```

Arguments

collection

Specifies the collection.

Description

The *get_object_name* command is a convenient way to get the *full_name* attribute of objects. When multiple objects are passed to the *get_attribute* command, the values are returned as a Tcl list, containing one entry for each object.

Examples

The following example shows how to use the *get_object_name* command:

```
pt_shell> get_object_name [get_cells i1]
i1
pt_shell> get_attribute [get_cells {i1 i2}] full_name
i1 i2
pt_shell> get_object_name [get_cells i*]
i1 i2 i3
```

See Also

- collections
- get_attribute

---

get_path_count_for_tag

Gets the number of paths for a given path tag set or multiple path tag sets.

Syntax

integer *get_path_count_for_tag*

```
[-name tag_names]
```

Feedback

Data Types

tag_names list

Arguments

-name *tag_names*

Specifies a list of path tag set names. The command returns the total number of paths in the listed path tag sets. Without this option, the command returns the total number of paths in all defined path tag sets.

Description

This command returns number of paths for a given path tag set or list of path tag sets, or all tag sets if you omit the *-name* option.

The command returns an integer, the sum of the numbers of paths in all the specified path tag sets. If a path belongs to multiple path tag sets, that path counts multiple times toward the reported total.

To report the number of paths separately for multiple path tag sets, use the *report_path_tag_set* command.

Examples

The following example tags a collection of timing paths and then reports the number of tagged paths in the path tag set.

```
pt_shell> set_app_var enable_path_tagging true
true
pt_shell> set mypaths [get_timing_paths ...]
pt_shell> create_path_tag_set $mypaths -name tag1
1
pt_shell> echo "Tagged [get_path_count_for_tag -name tag1] paths in tag1"
Tagged 100 paths in tag1
```

See Also

- get_timing_paths
- create_path_tag_set
- report_path_tag_set
- enable_path_tagging

---

get_path_group

Creates a collection of path groups from the current design. You can assign these path groups to a variable or pass them into another command.

Feedback

Syntax

collection *get_path_groups*

```
[-quiet]
[-regexp]
[-nocase]
[-filter expression]
patterns
```

Data Types

expression	string
patterns	list

Arguments

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, it modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

`-nocase`

When combined with the *-regexp* option, it makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-filter expression`

Filters the collection with the *expression* value. For any path groups that match the *patterns* values, the expression is evaluated based on the path group's attributes. If the expression evaluates to true, the path group is included in the result.

patterns

Matches path group names against patterns. Patterns can include the wildcard characters "*" and "?".

Description

The *get_path_groups* command creates a collection of path groups from the current design that match certain criteria. The command returns a collection if any path groups match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Path groups control the optimization cost function and also affect timing reports.

Feedback

You can use the *get_path_groups* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *get_path_groups* result to a variable.

When issued from the command prompt, the *get_path_groups* command behaves as though the *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_path_groups* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use *get_path_groups* as an argument to *query_objects*.

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

Examples

The following generates a timing report for each path group matching "Z*".

```
pt_shell> foreach_in_collection grp [get_path_groups Z*] { \\
?                report_timing -group $grp \\
?          }
```

See Also

- collections
- foreach_in_collection
- group_path
- query_objects
- report_path_group
- collection_result_display_limit

---

get_path_groups

Creates a collection of path groups from the current design. You can assign these path groups to a variable or pass them into another command.

Syntax

collection *get_path_groups*

```
[-quiet]
[-regexp]
```

Feedback

```
[-nocase]
[-filter expression]
patterns
```

Data Types

expression	string
patterns	list

Arguments

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, it modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

`-nocase`

When combined with the *-regexp* option, it makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-filter expression`

Filters the collection with the *expression* value. For any path groups that match the *patterns* values, the expression is evaluated based on the path group's attributes. If the expression evaluates to true, the path group is included in the result.

patterns

Matches path group names against patterns. Patterns can include the wildcard characters "*" and "?".

Description

The *get_path_groups* command creates a collection of path groups from the current design that match certain criteria. The command returns a collection if any path groups match the *patterns* and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

Path groups control the optimization cost function and also affect timing reports.

You can use the *get_path_groups* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *get_path_groups* result to a variable.

Feedback

When issued from the command prompt, the *get_path_groups* command behaves as though the *query_objects* had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_path_groups* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* options (for example, if you want to display the object class), use *get_path_groups* as an argument to *query_objects*.

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

Examples

The following generates a timing report for each path group matching "Z*".

```
pt_shell> foreach_in_collection grp [get_path_groups Z*] { \\
?                 report_timing -group $grp \\
?          }
```

See Also

- collections
- foreach_in_collection
- group_path
- query_objects
- report_path_group
- collection_result_display_limit

---

get_pg_pins

Create a collection of upf_pg_pin_info objects in the current design.

Syntax

string *get_pg_pins*

```
[-of_objects ell ]
```

Arguments

-of_objects

Gets the upf_pg_pin_info ojects of the object list. Of_objects takes cell list as value .

Feedback

Description

The *get_pg_pins* command creates a collection of upf_pg_pin_info objects in the current design. The command returns a collection handle (identifier) of upf_pg_pin_info objects oif *-of_objects* arguments. If no objects match the criteria, an empty string is returned.

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

Examples

The following example creates a supply net and creates collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_supply_net SN1 -domain TOP_DOMAIN
1
pt_shell> get_pg_pins -of_objects [get_cell FF1]
{"FF1/PWR FF1/GND"}
```

See Also

- create_supply_net
- report_supply_net

---

get_pin

Creates a collection of pins from the netlist. You can assign these pins to a variable or pass them into another command.

Syntax

collection *get_pins*

```
[-hierarchical]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-leaf]
[patterns | -of_objects objects]
```

Data Types

```
expression      string
patterns        list
objects         list
```

Feedback

Arguments

`-hierarchical`

Searches for pins level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the *find* UNIX command. For example, if there is a pin block1/adder/D[0], a hierarchical search finds it using "adder/D[0]". You cannot use the *-hierarchical* option with the *-of_objects* option.

`-filter` *`expression`*

Filters the collection with the *expression*. For any pins that match *patterns* (or *objects*), the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, this makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The*-exact* and *-regexp* options are mutually exclusive.

`-leaf`

Returns only leaf (non-hierarchical) pins. If you use the *-leaf* and *-of_objects* options together, and *-of_objects* specifies a collection of nets, the command returns pins on the leaf cells connected to the specified nets; in this case, hierarchical boundaries can be crossed to find these pins.

`patterns`

Matches pin names against patterns. The *patterns* option can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. The *patterns* option can also include collections of type pin. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

`-of_objects` *objects*

Creates a collection of pins connected to the specified objects. Each object is a named cell or net, cell collection or pin collection, a named library pin, or a collection of library pins. Note that library pins can not be combined with other types of objects. The*-of_objects* and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

Description

The *get_pins* command creates a collection of pins in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any pins match the *patterns* or *objects* options and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

When used with the *-of_objects* option, the *get_pins* command searches for pins connected to any cells or nets specified in the *objects*. For net objects, there are two variations of pins that will be considered. By default, only pins connected to the net at the same hierarchical level are considered. When combined with the *-leaf* option, only pins connected to the net that are on leaf cells are considered. In this case, hierarchical boundaries are crossed in order to find pins on leaf cells.

If no *patterns* (or *objects*) match any objects, and the current design is not linked, the design automatically links.

When a cell has bus pins, the *get_pins* command can find them in several ways. For example, if cell u1 has a bus "A" with indecies 2 to 0, and the bus_naming_style for your design is "%s[%d]", then to find these pins, you can use "u1/A[*]" as the pattern. You can also find the same three pins with "u1/A" as the pattern.

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_pins* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_pins* command result to a variable.

When issued from the command prompt, the *get_pins* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_pins* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the

Feedback

query_objects command, use the *get_pins* command as an argument to the *query_objects* command. For example, use this to display the object class.

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

Examples

The following example queries the 'CP' pins of cells that begin with 'o'. Although the output looks like a list, it is just a display.

```
pt_shell> get_pins o*/CP
{"o_reg1/CP", "o_reg2/CP", "o_reg3/CP", "o_reg4/CP"}
```

The following example shows that given a collection of cells, you can query the pins connected to those cells.

```
pt_shell> set csel [get_cells o_reg1]
{"o_reg1"}
pt_shell> query_objects [get_pins -of_objects $csel]
{"o_reg1/D", "o_reg1/CP", "o_reg1/CD", "o_reg1/Q", "o_reg1/QN"}
```

The following example shows the difference between getting local pins of a net and leaf pins of net. In this example, NET1 is connected to i2/a and reg1/QN. Cell i2 is hierarchical. Within i2, port a is connected to the U1/A and U2/A.

```
pt_shell> get_pins -of_objects [get_nets NET1]
{"i2/a", "reg1/QN"}
pt_shell> get_pins -leaf -of_objects [get_nets NET1]
{"i2/U1/A", "i2/U2/A", "reg1/QN"}
```

The following example shows how to create a clock using a collection of pins.

```
pt_shell> create_clock -period 8 -name CLK [get_pins o_reg*/CP]
1
```

See Also

- collections
- create_clock
- filter_collection
- get_cells
- link_design
- query_objects
- collection_result_display_limit

Feedback

get_pins

Creates a collection of pins from the netlist. You can assign these pins to a variable or pass them into another command.

Syntax

collection *get_pins*

```
[-hierarchical]
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-leaf]
[patterns | -of_objects objects]
```

Data Types

```
expression      string
patterns        list
objects         list
```

Arguments

`-hierarchical`

Searches for pins level-by-level relative to the current instance. The full name of the object at a particular level must match the patterns. The search is similar to the *find* UNIX command. For example, if there is a pin block1/adder/D[0], a hierarchical search finds it using "adder/D[0]". You cannot use the *-hierarchical* option with the *-of_objects* option.

`-filter expression`

Filters the collection with the *expression*. For any pins that match *patterns* (or *objects*), the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the pin is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, this makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The*-exact* and *-regexp* options are mutually exclusive.

`-leaf`

Returns only leaf (non-hierarchical) pins. If you use the *-leaf* and *-of_objects* options together, and *-of_objects* specifies a collection of nets, the command returns pins on the leaf cells connected to the specified nets; in this case, hierarchical boundaries can be crossed to find these pins.

`patterns`

Matches pin names against patterns. The *patterns* option can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. The *patterns* option can also include collections of type pin. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

`-of_objects` *`objects`*

Creates a collection of pins connected to the specified objects. Each object is a named cell or net, cell collection or pin collection, a named library pin, or a collection of library pins. Note that library pins can not be combined with other types of objects. The*-of_objects* and *patterns* options are mutually exclusive; you can specify only one. In addition, you cannot use the *-hierarchical* option with the *-of_objects* option.

Description

The *get_pins* command creates a collection of pins in the current design, relative to the current instance, that match certain criteria. The command returns a collection if any pins match the *patterns* or *objects* options and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

When used with the *-of_objects* option, the *get_pins* command searches for pins connected to any cells or nets specified in the *objects*. For net objects, there are two variations of pins that will be considered. By default, only pins connected to the net at the same hierarchical level are considered. When combined with the *-leaf* option, only pins connected to the net that are on leaf cells are considered. In this case, hierarchical boundaries are crossed in order to find pins on leaf cells.

If no *patterns* (or *objects*) match any objects, and the current design is not linked, the design automatically links.

When a cell has bus pins, the *get_pins* command can find them in several ways. For example, if cell u1 has a bus "A" with indecies 2 to 0, and the bus_naming_style for your design is "%s[%d]", then to find these pins, you can use "u1/A[*]" as the pattern. You can also find the same three pins with "u1/A" as the pattern.

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter *expression*; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_pins* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_pins* command result to a variable.

When issued from the command prompt, the *get_pins* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_pins* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command, use the *get_pins* command as an argument to the *query_objects* command. For example, use this to display the object class.

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

Examples

The following example queries the 'CP' pins of cells that begin with 'o'. Although the output looks like a list, it is just a display.

```
pt_shell> get_pins o*/CP
{"o_reg1/CP", "o_reg2/CP", "o_reg3/CP", "o_reg4/CP"}
```

The following example shows that given a collection of cells, you can query the pins connected to those cells.

```
pt_shell> set csel [get_cells o_reg1]
{"o_reg1"}
pt_shell> query_objects [get_pins -of_objects $csel]
{"o_reg1/D", "o_reg1/CP", "o_reg1/CD", "o_reg1/Q", "o_reg1/QN"}
```

The following example shows the difference between getting local pins of a net and leaf pins of net. In this example, NET1 is connected to i2/a and reg1/QN. Cell i2 is hierarchical. Within i2, port a is connected to the U1/A and U2/A.

Feedback

```
pt_shell> get_pins -of_objects [get_nets NET1]
{"i2/a", "reg1/QN"}
pt_shell> get_pins -leaf -of_objects [get_nets NET1]
{"i2/U1/A", "i2/U2/A", "reg1/QN"}
```

The following example shows how to create a clock using a collection of pins.

```
pt_shell> create_clock -period 8 -name CLK [get_pins o_reg*/CP]
1
```

See Also

- collections
- create_clock
- filter_collection
- get_cells
- link_design
- query_objects
- collection_result_display_limit

---

get_placement_blockages

Creates a collection of placement blockages from the current design.

Syntax

```
collection get_placement_blockages
[-all]
[-name expression]
```

Data Types

```
name          string
```

Arguments

`-all`

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

`-name`

Specifies the name of the placement blockage. It can be a wildcard expression

Feedback

- * (asterisk) indicates all placement blockages
- *xx** indicates any placement blockage whose name begins with *xx*
- PB_77 indicates one placement blockage that does not yet have a name and its object_id is 77
- PB_* indicates any placement blockage that does not yet have a name.

Description

This command creates a collection of placement blockages that meet the selection criteria. It returns a collection of placement blockages if one or more blockages meet the selection criteria. If no blockages match the selection criteria, it returns an empty string. You can use the *get_placement_blockages* command as an argument to another command or assign its result to a variable. Refer to the example below for details.

You can use the *get_placement_blockages* command at the command prompt, use it as an argument nested in another command, or assign its result to a variable.

See the *collections* man page for information about working with collections.

Examples

The following examples show some uses of this command to create placement blockages:

```
prompt> get_placement_blockages *
{"PB_1", "PB_2", "PB_3"}
```

See Also

- collections
- create_placement_blockage
- get_attribute
- query_objects

---

get_point_to_point_resistance

Return the equivalent resistance of a set of parasitic resistors from a given node to a given node in a net. The command returns a single value for a single corner case or a set of values for multi-corner case. Each value corresponds to the equivalent resistance for the corresponding corner.

Syntax

float *get_point_to_point_resistance*

Feedback

```
[-from node_name]
[-to   node_name]
[-quiet]
[-parasitic_corners corner_list]
[-all_parasitic_corners]
[-unit]
[-info]
[-ignore_layers]
```

Data Types

```
corner_list         list
node_name           string
node_name           string
```

Arguments

`-from node_name`

Constrains the selected resistors to be part of a path starting from the specified node. The node name can be either a pin, port, or an internal node, where the internal node name syntax is net_name:node_id, where the legal node_id range is from 1 to the number of nodes in the net. This option is used only with the -to option.

`-to node_name`

Constrains the selected resistors to be part of a path ending with the specified node, using the same node name syntax as the *-from* option.

`-quiet`

Suppresses warning and error messages if there are no objects that match. Syntax error messages are not suppressed.

`-parasitic_corners corner_list`

Indicates the parasitic corner ordering for resistance values. This option is only applicable for parasitics information obtained using *read_parasitics* with the GPD parasitic format. If the *-parasitic_corners* option is not specified, then the sin- gle parasitic corner specified by *set parasitic_corner_name* is returned.

`-all_parasitic_corners`

Specifies that all parasitic corners are to be returned. The parasitics ordering is determined according to the corner definitions of the GPD parasitics.

`-unit`

Show unit of get_point_to_point_resistance

`-info`

Get unit info of get_point_to_point_resistance

`-ignore_layers`

Specify layer to mark a certain connections to become ideal for calculations

Description

This command returns the equivalent point-to-point resistance (P2P) resistance between two points, which can be nodes/pins/ports. You must specify the *-from* and *-to* options to specify a path from which to get the equivalent resistance.

Examples

The following example queries the point to point resistance from internal node n22:1 to internal node n22:5

```
<_shell>get_point_to_point_resistance -from n22:1 -to n22:5
0.45
```

---

get_port

Creates a collection of ports from the current design or instance. You can assign these ports to a variable or pass them into another command.

Syntax

collection *get_ports*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[patterns]
[-of_objects objects]
```

Data Types

expression	string
patterns	list
objects	list

Arguments

`-filter expression`

Filters the collection with the *expression* option. For any ports that match the *patterns* option, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the cell is included in the result.

Feedback

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`patterns`

Matches port names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type port. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

`-of_objects` *`objects`*

Creates a collection of ports connected to the specified objects. Each object is a named net collection. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one.

Description

The *get_ports* command creates a collection of ports in the current design or instance that match certain criteria. The command returns a collection if any ports match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If the *port_search_in_current_instance* variable is true, then the *get_ports* command gets the ports of current instance. If the *port_search_in_current_instance* variable is *false* (the default), the *get_ports* command gets the ports of the current design.

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter the *expression* option; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You

can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_ports* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_ports* command result to a variable.

When issued from the command prompt, the *get_ports* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_ports* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the *query_objects* command, use the *get_ports* command as an argument to the *query_objects* command. For example, use this to display the object class.

For information about collections and the querying of objects, see the *collections* man page. In addition, refer to the *all_inputs* and *all_outputs* man pages, which also create collections of ports.

Examples

The following example queries all input ports beginning with 'mode'. Although the output looks like a list, it is just a display.

```
pt_shell> get_ports "mode*" -filter {direction == in}
{"mode[0]", "mode[1]", "mode[2]"}
```

The following example sets the driving cell for ports beginning with "in" to an FD2.

```
pt_shell> set_driving_cell -cell FD2 -library my_lib [get_ports in*]
```

The following example reports ports connected to nets matching the pattern "bidir*".

```
pt_shell> report_port [get_ports -of_objects [get_nets "bidir*"]]
```

See Also

- all_inputs
- all_outputs
- collections
- filter_collection
- query_objects

Feedback

- port_search_in_current_instance
- collection_result_display_limit

---

get_ports

Creates a collection of ports from the current design or instance. You can assign these ports to a variable or pass them into another command.

Syntax

collection *get_ports*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[patterns]
[-of_objects objects]
```

Data Types

expression	string
patterns	list
objects	list

Arguments

`-filter expression`

Filters the collection with the *expression* option. For any ports that match the *patterns* option, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the cell is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns. The *-regexp* and *-exact* options are mutually exclusive.

`-nocase`

When combined with the *-regexp* option, makes matches case-insensitive. You can use the *-nocase* option only when you also use the *-regexp* option.

`-exact`

Disables simple pattern matching. This is used when searching for objects that contain the "*" and "?" wildcard characters. The *-exact* and *-regexp* options are mutually exclusive.

`patterns`

Matches port names against patterns. Patterns can include the wildcard characters "*" and "?" or regular expressions, based on the *-regexp* option. Patterns can also include collections of type port. The *patterns* and *-of_objects* options are mutually exclusive; you can specify only one.

`-of_objects` *`objects`*

Creates a collection of ports connected to the specified objects. Each object is a named net collection. The *-of_objects* and *patterns* options are mutually exclusive; you can specify only one.

Description

The *get_ports* command creates a collection of ports in the current design or instance that match certain criteria. The command returns a collection if any ports match the *patterns* option and pass the filter (if specified). If no objects match the criteria, the empty string is returned.

If the *port_search_in_current_instance* variable is true, then the *get_ports* command gets the ports of current instance. If the *port_search_in_current_instance* variable is *false* (the default), the *get_ports* command gets the ports of the current design.

Regular expression matching is the same as in the *regexp* Tcl command. When using the *-regexp* option, take care in the way you quote the *patterns* and filter the *expression* option; use rigid quoting with curly braces around regular expressions. Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed.

You can use the *get_ports* command at the command prompt, or you can nest it as an argument to another command (for example, the *query_objects* command). In addition, you can assign the *get_ports* command result to a variable.

When issued from the command prompt, the *get_ports* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

The "implicit query" property of the *get_ports* command provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by

Feedback

the *query_objects* command, use the *get_ports* command as an argument to the *query_objects* command. For example, use this to display the object class.

For information about collections and the querying of objects, see the *collections* man page. In addition, refer to the *all_inputs* and *all_outputs* man pages, which also create collections of ports.

Examples

The following example queries all input ports beginning with 'mode'. Although the output looks like a list, it is just a display.

```
pt_shell> get_ports "mode*" -filter {direction == in}
{"mode[0]", "mode[1]", "mode[2]"}
```

The following example sets the driving cell for ports beginning with "in" to an FD2.

```
pt_shell> set_driving_cell -cell FD2 -library my_lib [get_ports in*]
```

The following example reports ports connected to nets matching the pattern "bidir*".

```
pt_shell> report_port [get_ports -of_objects [get_nets "bidir*"]]
```

See Also

- all_inputs
- all_outputs
- collections
- filter_collection
- query_objects
- port_search_in_current_instance
- collection_result_display_limit

---

get_power_domains

Create a collection of power domains in the current design.

Syntax

string *get_power_domains*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
```

Feedback

```
[patterns]
[-of_objects cells]
```

Data Types

```
expression              string
patterns                list
cells                   string
```

Arguments

`-filter expression`

Filters the collection with *expression*. For any power domains that match *patterns*, the expression is evaluated based on the power domain's attributes. If the expression evaluates to true, the power domain is included in the result. This option works the same as the *filter* command in dc_shell.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

`-nocase`

Makes matches case-insensitive.

`patterns`

Matches power domain names against patterns. Patterns can include the wildcard characters * (asterisk) and ? (question mark). For more details about using and escaping wildcards, refer to the *wildcards* man page. The *patterns* and *-of_objects* options are mutually exclusive.

`-of_objects cells`

Returns a collection of power domain that owns the cells. The *patterns* and *-of_objects* options are mutually exclusive.

Description

The *get_power_domains* command creates a collection of power domains in the current design, that match certain criteria. The command returns a collection handle (identifier) if any power domains match the *patterns* or *-of_objects* value and pass the filter, if specified. If no objects match the criteria, the empty string is returned.

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

Feedback

Examples

The following example queries the power domain and creates collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_power_domain SUB_DOMAIN -power_down \\
           -power_down_ctrl [get_nets pd_ctrl] \\
           -object_list [get_cells mid1]
1
pt_shell> get_power_domain {TOP_DOMAIN SUB_DOMAIN}
{"", ""}

prompt> get_power_domain TOP*
{""}
```

See Also

- report_power_domain

---

get_power_group_objects

Returns a collection of cells in a power group.

Syntax

collection *get_power_group_objects*

group_names

Data Types

group_names list

Arguments

group_names

Specifies the power groups of which the collection of cells will be returned.

Description

The *get_power_group_objects* command returns a collection of cells contained by the specified power groups.

Examples

In the following example, the cells included in the newly created power group are displayed.

Feedback

```
pt_shell> create_power_group -name my_group [get_cells "u1 u2"]
1
pt_shell> get_power_group_objects my_group
{"u1", "u2"}
```

See Also

- create_power_group
- remove_power_groups
- report_power_groups

---

get_power_per_pgpin

Reports power on every power pin of the cell

Syntax

```
collection get_power_per_pgpin
  [-power_type switching | internal | leakage | total]
  [-return_supply_level leaf | top | through_switch]
  object_list
```

Data Types

```
object_list      list
```

Arguments

`-power_type switching | internal | leakage | total`

Specifies the power type to be reported.

`-return_supply_level leaf | top | through_switch`

Specifies the power net to be reported:

- *leaf* - Reports the hierarchical net connected to the leaf cell.
- *top* - Report the flat net connecting the leaf cell from the top most context.
- *through_switch* - Reports the supply_net after getting propagated through the switches.

object_list

Specifies a list of leaf cells.

Feedback

Description

The *get_power_per_pgpin* command reports the power on the power pins of the leaf cell. The command output is in the form of a list of lists where every power pin list consists of the power pin name, supply_net or rail name and the power associated with the power pin.

The variable power_enable_multi_rail_analysis must be set to true for *get_power_per_pgpin* command to function properly.

Examples

In the following example, the *get_power_per_pgpin* command is specified after *update_power*.

```
pt_shell> get_power_per_pgpin U76
{U76 {VDD 'VDD' 4.044e-07} {VNW '' 6.795e-10} {VPW '' 0}}
```

See Also

- get_power_per_rail
- report_power
- report_power_pin_info

---

get_power_per_rail

Reports power on given rails or supply_nets of the design for a given list of hierarchical cells.

Syntax

```
collection get_power_per_rail
  [-rails rail_list]
  [-power_type switching | internal | leakage | total]
  [-return_supply_level leaf | top | through_switch]
  object_list
```

Data Types

```
rail_list        list
object_list      list
```

Arguments

`-rails rail_list`

Specifies the power nets/rails to be reported.

`-power_type switching | internal | leakage | total`

Specifies the power type to be reported.

Feedback

`-return_supply_level leaf | top | through_switch`

Specifies the power net to be reported:

- *leaf* - Reports the hierarchical net connected to the leaf cell.
- *top* - Report the flat net connecting the leaf cell from the top most context.
- *through_switch* - Reports the supply_net after getting propagated through the switches.

object_list

Specifies a list of hierarchical/leaf cells.

Description

The *get_power_per_rail* command reports the power on the rails for each of the cells specified. The command output is in the form of a list of lists where every list consists of the supply_net or rail name , hierarchy or cell name and the associated power with the rail for the given cell. Supports when multi-rail analysis is enabled. Please set variable "power_enable_multi_rail_analysis true".

Examples

In the following example, the *get_power_per_rail* command is specified after *update_power*.

```
pt_shell> get_power_per_rail "four_mini_inst/two_mini_1" -rails
 four_mini_inst/VDD_MINI_1
{ four_mini_inst/VDD_MINI_1 four_mini_inst/two_mini_1 9.354e-05 }
  { unconnected four_mini_inst/two_mini_1 1.031e-05 }

pt_shell> get_power_per_rail "four_mini_inst" -power_type total \\
          -return_supply_level through_switch
{ unconnected four_mini_inst 3.063e-05 } { VDD four_mini_inst 0.0001921 }
```

See Also

- get_power_per_pgpin
- report_power
- report_power_pin_info

---

get_power_switches

Create a collection of UPF power switches in the current design.

Syntax

string *get_power_switches*

Feedback

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-of_objects upf_power_domain|upf_supply_net]
[patterns]
```

Data Types

```
expression      string
patterns        list
```

Arguments

`-filter expression`

Filters the collection with the *expression* argument. For any UPF power switches that match the *patterns* argument, the expression is evaluated based on the power switch's attributes. If the expression evaluates to true, the power switch is included in the result. This option works the same as the *filter* command in pt_shell.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

`-nocase`

Makes matches case-insensitive.

patterns

Matches supply net names against patterns. Patterns can include the wildcard characters "*" (asterisk) and "?" (question mark). For more details about using and escaping wildcards, see the *wildcards* man page. The *patterns* and *-of_objects* arguments are mutually exclusive.

-of_objects

Gets the power switches of the object list. Objects could be combination of upf_power_domain or upf_supply_net. The *patterns* and *-of_objects* arguments are mutually exclusive.

Description

The *get_power_switches* command creates a collection of UPF power switch objects in the current design, that match certain criteria. The command returns a collection handle

Feedback

(identifier) if any supply nets match the *patterns* or *-of_objects* arguments and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

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

Examples

The following example creates a power switch and creates collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_supply_net SN1 -domain TOP_DOMAIN
1
pt_shell> create_supply_net SN2 -domain TOP_DOMAIN
1
pt_shell> create_power_switch my_switch -domain TOP_DOMAIN \\
          -output_supply_port {OUT SN2} \\
          -input_supply_port {IN SN2} \\
          -on_state {state1 IN {CNTL} } \\
          -control_port {ctrl CNTL}

pt_shell> get_power_switches *
{"my_switch"}
```

See Also

- create_power_switch
- report_power_switch

---

get_qtm_ports

Creates a collection of quick timing model (QTM) ports. You can assign these QTM ports to a variable or pass them into another command.

Syntax

collection *get_qtm_ports*

```
        patterns
```

Data Types

patterns	list

Feedback

Arguments

patterns

Matches QTM port names against patterns. Patterns can include the wildcard characters "*" and "?".

Description

The *get_qtm_ports* command creates a collection of QTM ports from the current QTM model that match certain criteria. The command returns a collection if any QTM ports match the *patterns* option. If no objects match the criteria, an empty string is returned.

To create a QTM port, use the *create_qtm_port* command.

See the *collections* man page for information about collections and the querying of objects.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following command applies the *set_qtm_port_load* command to all QTM ports in the model matching "IN*".

```
pt_shell> set_qtm_port_load -value 2.0 [get_qtm_ports "IN*"]
```

The following command creates a delay arc from QTM port A to all QTM ports in the model matching "OUT*".

```
pt_shell> create_qtm_delay_arc -from A -to [get_qtm_ports "OUT*"] -value
 3.0
```

See Also

- create_qtm_constraint_arc
- create_qtm_delay_arc
- create_qtm_model
- create_qtm_port
- report_qtm_model
- save_qtm_model
- set_qtm_port_drive
- set_qtm_port_load

Feedback

get_random_numbers

Generates a list of random numbers for a specified variation.

Syntax

```
list get_random_numbers
  -sample_size size_value
  -seed seed_value
  variation_object
```

Data Types

size_value	integer
seed_value	integer
variation_object	collection

Arguments

`-sample_size` *size_value*

Specifies the sample size.

`-seed` *seed_value*

Specifies the seed for this particular stream. This number must be a positive integer. A different number generates a different sample.

variation_object

Specifies the variation from which to generate random samples. There can only be one object.

Description

This command generates random numbers in a list.

Examples

In the following example, the returned list contains a sample of size 500 of random numbers generated from a variation with a normal distribution with mean 15.0 and standard deviation 2.0. The seed for this random number stream is 17777.

```
pt_shell> create_variation -name test -type normal -values { 15.0 2.0 }
_sel21

pt_shell> set rnd_list [get_random_numbers -sample_size 500 \\
          -seed 17777 [get_variations test]]
10.613445  14.553537  13.051448  16.382267 ...
```

Feedback

get_related_supply_net

Returns a collection of UPF related supply nets for given pins.

Syntax

```
collection get_related_supply_net
  [-ground]
  [pins]
```

Data Types

```
pins        list
```

Arguments

`-ground`

Includes ground supply in the collection.

pins

Specifies a list of pins to be queried.

Description

The *get_related_supply_net* command returns a collection of UPF supply nets for the given pins. Power supply net is returned by default. If the ground options is specified, ground supply net is returned.

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

Examples

The following example returns the related supply net of all the pins of instance LS.

```
prompt> get_related_supply_nets [get_pin Mod1/LS/*]
{"SS1.pwr" "SS1.pwr"}
```

See Also

- report_supply_net

get_resistors

Creates a collection of parasitic resistors from the specified net. You can assign these resistors to a variable or pass them into another command.

Syntax

collection *get_resistors*

Feedback

```
[-filter expression]
[-quiet]
[-parasitic_corners corner_list]
[-all_parasitic_corners]
[-of_objects objects]
[-from node_name]
[-to node_name]
[-unit]
[-shortest]
[-info]
```

Data Types

```
expression          string
corner_list         list
node_name           string
objects             list
node_name           string
```

Arguments

`-filter expression`

Filters the collection with the *expression* value. For any resistors that match *patterns* or *objects*, the expression is evaluated based on the resistor's attributes. If the expression evaluates to true, the resistor is included in the result.

`-quiet`

Suppresses warning and error messages if there are no objects that match. Syntax error messages are not suppressed.

`-parasitic_corners corner_list`

Indicates the parasitic corner ordering for resistance values. This option is only applicable for parasitics information obtained using *read_parasitics* with the GPD parasitic format. If the *-parasitic_corners* option is not specified, then the single parasitic corner specified by *set parasitic_corner_name* is returned.

`-all_parasitic_corners`

Specifies that all parasitic corners are to be returned. The parasitics ordering is determined according to the corner definitions of the GPD parasitics.

`-of_objects objects`

Creates a collection of resistors associated with the specified objects. Each object must be a named net or net collection.

You must use either the *-of_objects* option or the *-from* and *-to* options.

`-from` *`node_name`*

Constrains the selected resistors to be part of a path starting from the specified node. The node name can be either a pin, port, or an internal node, where the internal node name syntax is *net_name*:*node_id*, where the legal *node_id* range is from 1 to the number of nodes in the net. This option is used only with the *-to* option.

`-to` *`node_name`*

Constrains the selected resistors to be part of a path ending with the specified node, using the same node name syntax as the *-from* option.

`-unit`

Show unit of get_resistors

`-shortest`

Get the shortest path of the selected resistors with the specified starting node and the specified ending node. This option is used only the *-from* and *-to* options.

`-info`

Get unit info of get_resistors

Description

This command creates a collection of resistors for the specified nets that match certain criteria. You must use either the *-of_objects* option and specify a net collection or the *-from* and *-to* options to specify a path from which to get the coupling capacitors. If no objects match the criteria, an empty string is returned.

When issued from the command prompt, the *get_resistors* command behaves as though the *query_objects* command had been called to display the objects in the collection. By default, a maximum of 100 objects is displayed; you can change this maximum using the *collection_result_display_limit* variable.

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

Examples

The following example queries the parasitic resistors of net n22, subject to those objects whose max corner resistance is greater than 0.5e-3.

```
pt_shell> get_resistors -of_objects n22 -filter "resistance_max > 0.5e-3"
{"resistor"}
```

Feedback

See Also

- collections
- filter_collection
- get_ground_capacitors
- get_coupling_capacitors
- link_design
- query_objects
- collection_result_display_limit

---

get_rule_property

Gets a property value on a rule. Properties affect how a specific rule check is performed, or how the output of a rule violation is presented.

Syntax

string *get_rule_property*

```
property_name
rule
```

Data Types

```
property_name     string
rule              string
```

Arguments

property_name

The name of a valid property on the specified rule.

rule

Specifies the name of an existing rule.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Obtains a property value for one rule. Properties can control rule checking; for example, the limit of some comparison. Properties can also control the output; for example, the maximum number of pins shown in the more info section of a violation.

Feedback

Examples

The following example gets the *max_percent* property for rule *EXD_0009* and assigns it to a variable.

```
pt_shell> set maxPercentVar [get_rule_property max_percent EXD_0009]
```

See Also

- create_rule

---

get_rules

Creates a collection of rules. You can assign the resulting collection to a variable or pass it into another command.

Syntax

collection *get_rules*

```
[-filter expression]
[-quiet]
patterns | -of_objects objects
```

Data Types

```
expression        string
patterns          list
objects           list
```

Arguments

`-filter expression`

Filters the collection with *expression*. For any rules that match *patterns* (or *objects*) the expression is evaluated based on the rule's attributes. If the expression evaluates to true, the rule is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-of_objects objects`

Creates a collection of rules contained in the specified rulesets. In this case, each entry is either a named ruleset or a collection of rulesets. *-of_objects* and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches rule names against patterns. Patterns can include the wildcard characters "*" and "?". Patterns can also include collections of type rule. *patterns* and *-of_objects* are mutually exclusive; you must specify one, but not both.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

The *get_rules* command creates a collection of rules defined in the session that match certain criteria. The command returns a collection if any rules match the *patterns* or *objects* and pass the filter (if specified). If no rules match the criteria, the empty string is returned.

Examples

The following example performs the *disable_rule* command on the collection of rules matching pattern "CSL_*".

```
pt_shell> disable_rule [get_rules CSL_*]
```

See Also

- create_rule
- create_ruleset
- get_rulesets

---

get_rulesets

Creates a collection of rulesets. You can assign the resulting collection to a variable or pass it into another command.

Syntax

collection *get_rulesets*

```
[-filter expression]
[-quiet]
patterns | -of_objects objects
```

Data Types

```
expression      string
patterns        list
objects         list
```

Feedback

Arguments

`-filter` *expression*

Filters the collection with *expression*. For any rulesets that match *patterns* (or *objects*) the expression is evaluated based on the ruleset's attributes. If the expression evaluates to true, the ruleset is included in the result.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-of_objects` *objects*

Creates a collection of rulesets containing the specified rules. In this case, each entry is either a named rule or a collection of rules. *-of_objects* and *patterns* are mutually exclusive; you must specify one, but not both.

patterns

Matches ruleset names against patterns. Patterns can include the wildcard characters "*" and "?". Patterns can also include collections of type ruleset. *patterns* and *-of_objects* are mutually exclusive; you must specify one, but not both.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

The *get_rulesets* command creates a collection of rulesets defined in the session that match certain criteria. The command returns a collection if any rulesets match the *patterns* or *objects* and pass the filter (if specified). If no rulesets match the criteria, the empty string is returned.

Examples

The following example performs the *disable_rule* command on the rules in ruleset "my_set1"

```
pt_shell> disable_rule [get_rulesets my_set1]
```

See Also

- create_rule
- create_ruleset
- get_rules

Feedback

get_selection

Returns a collection containing the current selection in the GUI.

Syntax

```
collection get_selection
[-slct_targets target_selection_bus
  [-slct_targets_operation operation]]
-create_slct_buses
[-name selection_bus]
[-type object_type]
[-design design]
[-more_than more]
[-fewer_than fewer]
[-count]
[-num num]
[-type_list]
```

Data Types

target_selection_bus	string
operation	string
selection_bus	string
object_type	string
design	string
more	integer
fewer	integer
num	integer

Arguments

`-slct_targets` *target_selection_bus*

Specifies the name of a selection bus in which to store the result. When you use this option, the *target_selection_bus* value is also returned as result of the command. By default, the result is returned in a collection.

`-slct_targets_operation` *operation*

Specifies which operation should be used when storing the result in the selection bus specified by the *target_selection_bus* argument. The valid values for the *operation* argument are *clear*, *add*, and *remove*. You can use the *-slct_targets_operation* option only if you also use the *-slct_targets* option.

`-create_slct_buses`

Specifies to create a new selection bus in which to store the result. Using this option is equivalent to using the *create_selection_bus* command to create a selection bus and then providing the created selection bus to the *-slct_targets* option.

Feedback

`-name selection_bus`

Specifies the selection bus from which the selection is taken.

`-type object_type`

Specifies the type of selected object with which to filter the selection. The valid values for *object_type* and their descriptions are as follows:

```
design -design object
port -port object
net -net object
cell -cell object
pin -pin object
path -timing path object
```

`-design design`

Returns only objects in the filter specified by the value of *design*.

`-more_than more`

Returns true if the selection bus contains more than the number of objects specified by *more*.

`-fewer_than fewer`

Returns true if the selection bus contains more than the number of objects specified by *fewer*.

`-count`

Returns the number of elements contained in the selection bus.

`-num num`

Returns only the number of objects specified by *num* or fewer.

`-type_list`

Returns a Tcl list of the types of elements contained in the selection bus.

Description

This command returns the current selection in the tool as a collection. It returns a collection handle (identifier) if any objects are selected. If no objects are selected, the command returns an empty string. If you do not use the *-type* option, the command returnS a heterogeneous collection of all objects currently selected. If you use *-type*, the collection is filtered to a homogeneous one containing only objects of the type you specify.

You can use the *get_selection* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*), or assign the *get_selection* result to a variable.

Feedback

When issued from the command prompt, *get_selection* behaves as if you had called *query_objects* to display the objects in the collection. By default, the command displays a maximum of 100 objects. You can change this maximum by using the *collection_result_display_limit* variable.

The "implicit query" property of *get_selection* provides a fast, simple way to display cells in a collection. However, if you want the flexibility provided by the options of the *query_objects* command (for example, if you want to display the object class), use *get_selection* as an argument to *query_objects*.

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

Examples

The following example returns the collection of the selected cell objects.

```
prompt> get_selection -type cell
{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
```

The following example assigns the collection to a variable named *collect_a* that is passed to the *query_objects* command.

```
prompt> set collect_a [get_selection -type cell]
{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
prompt> query_objects $collect_a
{"I_PRGRM_CNT_TOP", "I_DATA_PATH", "I_REG_FILE", "I_STACK_TOP"}
```

See Also

- change_selection
- collections
- filter_collection
- query_objects

---

get_shapes

Creates a collection by selecting shapes from the design.

Syntax

```
collection get_shapes
[-of_objects objects]
```

Data Types

```
objects        collection
```

Feedback

Arguments

`-of_objects` *of_objects*

Creates a collection containing the shapes connected to the specified net. Each object in the list is a collection.

This command creates a collection of shapes by selecting shapes from the current design that meet the selection criteria. It returns a collection handle if one or more shapes meet the selection criteria. If no shapes match the selection criteria, it returns an empty string.

Use the *get_shapes* command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the collection command man page for information about working with collections.

Examples

The following example gets all shapes connected to the specified net.

```
prompt> get_shapes * -of_objects [get_nets net1]
{PATH_31_0 PATH_31_1}
```

See Also

- collections
- get_attribute
- query_objects

---

get_si_bottleneck_nets

Creates a collection of nets that have the largest crosstalk effects that contribute to timing violations.

Syntax

status *get_si_bottleneck_nets*

```
[-cost_type type]
[-slack_lesser_than slack_limit]
[-minimum_active_aggressors active_aggressor_count]
[-delta_delay_threshold threshold]
[-min]
[-max]
[-all_nets]
[-nets net_list]
```

```
[-max_nets number]
[-include_clock_nets]
```

Data Types

type	list
slack_limit	float
number	integer
digits	integer
active_aggressor_count	integer
net_list	list

Arguments

`-cost_type` *type*

Sorts the nets based on the cost. The *delta_delay*, *delta_delay_ratio*, *total_victim_delay_bump* options are the costs of the victim nets. The *delay_bump_per_aggressor* option for aggressor nets of the selected victim nets.

`-slack_lesser_than` *slack_limit*

Applies the cost function only to victim nets with slack less than the specified limit. The default is zero. Specify a positive value to find more bottleneck nets, and vice versa.

`-minimum_active_aggressors` *active_aggressor_count*

Sets the minimum number of active aggressors for the net to be selected. The default is 1.

`-delta_delay_threshold` *threshold*

Restricts to only the victim nets whose delta delay cost exceeds the specified threshold. The threshold must be at least zero. The default is zero. This option applies only if the *-cost_type* option is set to *delta_delay*.

`-min`

Considers only minimum-delay (hold) constraints.

`-max`

Considers only maximum-delay (setup) constraints.

`-all_nets`

Considers all nets with SI cost that contribute to negative path slacks. By default, the command reports the 20 nets with the worst bottleneck cost.

`-nets` *list*

Restricts the bottleneck analysis to only the nets specified in a list or collection of nets. By default, the command considers all nets in the design.

Feedback

`-max_nets` *number*

Sets the maximum number of nets to be collected. The default is 20.

`-include_clock_nets`

Includes the clock nets for bottleneck analysis. By default, clock nets are excluded.

Description

This command has the same behavior as the *report_si_bottleneck* command, but returns nets as a collection instead of reporting on them. For more information, see the man page for the *report_si_bottleneck* command.

Examples

The following command creates a collection of nets belonging to paths with setup or hold violations that have the largest delta delays.

```
pt_shell> set my_nets [get_si_bottleneck_nets]
{"DX23[17]", "REG_ADDR[2]", "REG_DATA[5]", "RE_RDY", "REG_DATA[6]",
"REG_ADDR[4]", "DX23[22]", "DX23[11]", "REG_ADDR[7]", "DX23[1]"}

pt_shell> foreach_in_collection n $my_nets \\
  {echo [get_attribute [get_nets $n] si_xtalk_bumps]}
{ n24331 0.025178  0.025697 }
{ n26710 0.021577  0.022355 } { RE_RDY screened_by_macro_model
 screened_by_macro_model }
{ _CG_RDY_IN 0.022645  0.023179 }
{ REG_ADDR[2] 0.018784  0.018983 }
{ REG_ADDR[8] 0.014359  0.014527 }
 ...
```

See Also

- report_si_bottleneck
- set_coupling_separation
- size_cell

---

get_sms_scenarios

Gets a voltage/corner scenario collection for use with SMS features.

Syntax

```
status get_sms_scenarios
  [-supply_group supply_group_list]
  [-reference_names levels]
  [-propagated]
```

Feedback

Data Types

```
supply_group_list     list
levels                list
```

Arguments

`-supply_group` *supply_group_list*

Specifies a supply group. The reference voltage names for the supply group should be previously specified using the *set_voltage_levels* command. Multiple *-supply_group* options can be applied to the command. For every *-supply_group* option, there must be a corresponding *-reference_names* option. The reference voltage names of the *supply_group* will be part of the scenario being specified.

`-reference_names` *levels*

Specifies one or more reference voltage names of the corresponding *supply_group* that was specified using the *-supply_group* option. Multiple *-reference_names* options can be applied to the command. For every *-reference_names* option, there must be a corresponding *-supply_ggroup* option. The reference voltage names of the *supply_group* will be part of the scenario being specified.

`-propagated`

With this option the command returns a collection including all SMS scenarios used for comprehensive Multi Voltage timing analysis of the design. The collection includes only the scenarios relevant to all the analyzed timing paths. Voltage configurations which are not relevant to any timing path are not included in this collection. This option is only available with SMVA analysis and after update_timing. This option is mutually exclusive with command options *-supply_group* and *-reference_names*.

`-disabled`

With this option the command returns a collection including all SMS scenarios disabled using *configure_sms_scenarios -disable*. The same collection can be reported using *report_sms_scenarios -disabled*.

Description

The command specifies a voltage scenario or a set of voltage scenarios. For each *-supply_group* specified, there should be corresponding *-reference_names* specified. The reference voltage names for the *supply_group* become part of the scenario. If any *supply_group* is not referenced, it is assumed that all reference voltage names for the *supply_group* are part of the scenario.

In order to reference a supply group, the command *set_voltage_levels* must previously have been run. *set_voltage_levels* is used to specify the names of the reference voltage names.

After you run the *get_sms_scenarios* command and reference a particular supply group, it is not possible to reset the reference voltage names of the supply group using *set_voltage_levels*, unless the design is reset using *reset_design*.

When specifying multiple supply groups, be sure to specify the options in the correct order. The tool determines correspondence of the *-supply_group* and *-reference_names* options by order. The first *-supply_group* option is matched with the first *-reference_names* option, and so on.

Examples

The following example specifies reference voltage names for three supply groups using the *set_voltage_levels* command, then creates several scenarios.

Scenario *$sc1* includes level *sleep* of supply group *A* and levels *1.0V,1.1V* of supply group *B*, and all levels of supply group *C*. Scenario *$sc2* includes level *1.2V* of supply group *B*, and all levels of groups *A* and *C*. Both *$sc1* and *$sc2* represent several voltage scenarios. Scenario *$sc3* includes level *fast* of supply group *A*, level *1.2V* of supply group *B*, and level *v0.9* of supply group *C*. Scenario *$sc3* specifies a single voltage scenario, because all supply groups are specified, and only one level is in the scenario for each supply group. Note that scenario *$sc3* is included within the *$sc2* scenarios.

```
pt_shell> set_voltage_levels -reference_names {sleep slow fast}
 [get_supply_group A]
1
pt_shell> set_voltage_levels -reference_names {1.0V 1.1V 1.2V}
 [get_supply_group B]
1
pt_shell> set_voltage_levels -reference_names {v0.9 v1.0}
 [get_supply_group C]
1
pt_shell> set sc1 [get_sms_scenarios -supply_group A -reference_names
 {sleep} \\
          -supply_group B -reference_names {1.0V 1.1V}]
__sel22
pt_shell> set sc2 [get_sms_scenarios -supply_group B -reference_names
 {1.2V}]
__sel22
pt_shell> set sc3 [get_sms_scenarios -supply_group A -reference_names
 {fast} \\
          -supply_group B -reference_names {1.2V} \\
          -supply_group C -reference_names {v0.9}]
__sel22
```

See Also

- reset_design
- set_voltage_levels

Feedback

get_supply_groups

Creates a collection of supply groups in the current design.

Syntax

string *get_supply_groups*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[patterns]
```

Data Types

expression	string
patterns	list

Arguments

`-filter expression`

Filters the collection with the *expression* option. For any supply groups that match the *patterns* option, the expression is evaluated based on the supply group's attributes. If the expression evaluates to true, the supply group is included in the result. This option works the same as the *filter* command in dc_shell.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

`-nocase`

Makes matches case-insensitive.

patterns

Matches supply group names against patterns. Patterns can include the wildcard characters "*" (asterisk) and "?" (question mark). For details about using and escaping wildcards, refer to the *wildcards* man page. The *patterns* and *-of_objects* arguments are mutually exclusive.

Feedback

Description

The *get_supply_groups* command creates a collection of supply groups in the current design that match certain criteria. The command returns a collection handle (identifier) if any supply groups match the *patterns* or *-of_objects* arguments and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

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

Examples

The following example creates a collection of supply_groups.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_supply_net SN1 -domain TOP_DOMAIN
1
pt_shell> get_supply_nets {SN1}
{"SN1"}

prompt> get_supply_groups SN*
{"SN1"}
```

See Also

- report_supply_group

---

get_supply_nets

Create a collection of UPF supply nets in the current design.

Syntax

string *get_supply_nets*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-hierarchy]
[-pg_types power|ground|pwell|nwell]
[-segments ]
[-of_objects Cell|upf_pg_pin_info|upf_power_domain]
[patterns]
```

Data Types

```
expression      string
patterns        list
```

Feedback

Arguments

`-filter` *`expression`*

Filters the collection with the *expression* option. For any UPF supply nets that match the *patterns* option, the expression is evaluated based on the supply net's attributes. If the expression evaluates to true, the supply net is included in the result. This option works the same as the *filter* command in dc_shell.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Uses the *patterns* argument as real regular expressions rather than simple wildcard patterns.

`-nocase`

Makes matches case-insensitive.

`-hierarchy`

Searches for supply nets in the current scope and all of its descendant scopes. If you specify this option, you can specify only simple names in the patterns argument; the name cannot contain hierarchy delim- iter characters. If you do not specify this option, the tool searches for supply nets only in the current scope. You can use hierarchy delimiter characters in the patterns argument to specify hierarchical sup- ply nets relative to the current scope.

`-pg_types`

Filters supply net based on pg_types. pg_types could be list of following values: power, ground, pwell, nwell.

`-segments`

Gets all the net segments connected to the list of nets. The search will stop at power switches.

`-of_objects`

Gets the supply net of the object list. Objects could be combination of cell, upf_power_domain or upf_pg_pin_info. get_pg_pins returns a collection of upf_pg_pin_info objects which could be used as of_objects. The *patterns* and *-of_objects* arguments are mutually exclusive.

`patterns`

Matches supply net names against patterns. Patterns can include the wildcard characters "*" (asterisk) and "?" (question mark). For details about using

Feedback

and escaping wildcards, refer to the *wildcards* man page. The *patterns* and *-of_objects* arguments are mutually exclusive.

Description

The *get_supply_nets* command creates a collection of UPF supply nets in the current design that match certain criteria. The command returns a collection handle (identifier) if any supply nets match the *patterns* or *-of_objects* arguments and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

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

Examples

The following example creates a supply net and creates collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_supply_net SN1 -domain TOP_DOMAIN
1
pt_shell> get_supply_nets {SN1}
{"SN1"}

prompt> get_supply_nets SN*
{"SN1"}
```

See Also

- create_supply_net
- report_supply_net

---

get_supply_ports

Create a collection of UPF supply ports in the current design.

Syntax

string *get_supply_ports*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[-of_objects objects]
[patterns]
```

Feedback

Data Types

expression	string
patterns	list

Arguments

`-filter` *expression*

Filters the collection with the *expression* option. For any UPF supply ports that match the *patterns* option, the expression is evaluated based on the supply port's attributes. If the expression evaluates to true, the supply port is included in the result. This option works the same as the *filter* command in dc_shell.

`-quiet`

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

`-regexp`

Uses the *patterns* option as real regular expressions rather than simple wildcard patterns.

`-nocase`

Makes matches case-insensitive.

`-of_objects objects`

Creates a collection of supply ports connected to the specified objects. The -of_objects and patterns options are mutually exclusive; you can specify only one. In addition, you cannot use the -hierarchical option with the -of_objects option.

patterns

Matches supply port names against patterns. Patterns can include the wildcard characters "*" (asterisk) and "?" (question mark). For details about using and escaping wildcards, refer to the *wildcards* man page. The *patterns* and *-of_objects* options are mutually exclusive.

Description

The *get_supply_ports* command creates a collection of UPF supply ports in the current design that match certain criteria. The command returns a collection handle (identifier) if any supply ports match the *patterns* or *-of_objects* options and pass the filter (if specified). If no objects match the criteria, an empty string is returned.

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

Feedback

Examples

The following example creates a supply port and creates a collection of it.

```
pt_shell> create_power_domain TOP_DOMAIN
1
pt_shell> create_supply_port SN1 -domain TOP_DOMAIN
1
pt_shell> get_supply_ports {SN1}
{"SN1"}

prompt> get_supply_ports SN*
{"SN1"}
```

See Also

- create_supply_port

---

get_supply_sets

Creates a collection of UPF supply sets in the current design.

Syntax

collection *get_supply_sets*

```
[-filter expression]
[-quiet]
[-regexp]
[-nocase]
[patterns]
```

Data Types

expression	string
patterns	list

Arguments

`-filter expression`

Filters the collection with the *expression* option. For any UPF supply set that matches the *patterns* option, the expression is evaluated based on the supply set's attributes. If the expression evaluates to true, the supply set is included in the result.

This option works the same as the *filter* command in dc_shell.

`-quiet`

Suppresses warning and error messages if no object matches. Syntax error messages are not suppressed.

Feedback

`-regexp`

Uses the *patterns* option as real regular expressions rather than simple wildcard patterns.

`-nocase`

Performs case-insensitive matching.

`patterns`

Matches supply set names against patterns. Patterns can include the asterisk (*) and question mark (?) wildcard characters. For information about using and escaping wildcards, see the *wildcards* man page.

Description

The *get_supply_sets* command creates a collection of UPF supply sets in the current design that match certain criteria. The command returns a collection handle (identifier) if any supply set matches the *patterns* options and pass the filter (if specified). If no object matches the criteria, an empty string is returned.

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

Examples

The following example creates a supply set and creates a collection of it.

```
prompt> create_supply_set SS1
1
prompt> get_supply_sets {SS1}
{"SS1"}
prompt> get_supply_sets SS*
{"SS1"}
```

See Also

- create_supply_set

---

get_switching_activity

Gets switching activity annotation on nets, pins, ports, and cells of the current design.

Syntax

integer *get_switching_activity*

```
[-state_condition state]
[-path_sources name_list]
[-toggle_rate]
[-glitch_rate]
```

```
[-static_probability]
[-rise | -fall]
[-only_related_clock clock_name]
[-toggle_limit limit]
[-exclude exclusion_group]
[-include_only inclusion_group]
[-average_activity]
[-relative_toggle_rate]
object_list
```

Data Types

state	string
name_list	string
clock_name	string
limit	integer
exclusion_group	string
inclusion_group	string
object_list	list

Arguments

`-state_condition` *state*

Specifies the state condition when getting state-dependent toggle rate and glitch rate on pins or state-dependent static probabilities on cells. State dependent toggle rate and glitch rate can be retrieved for a pin when the internal power of the library cell pin is characterized with state-dependent power tables. State-dependent static probabilities for a cell is annotated when the cell leakage power is characterized with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal/leakage power characterization. The state condition should be enclosed between " ". Moreover, if you want to get switching activity information for the default state condition, then the *state* argument should be "default". If the *-state_condition* option is applied to a cell, then only static probability is reported. Only leakage power is associated with cells. If the *state* argument is applied to a pin, then the tool looks for the condition in the internal power tables and report the toggle rate/glitch rate/static probability for that pin.

Given a design object, the command returns the list comprised of object name, the *state* argument, static probability for the *state* argument and the annotation source indicator. If the *state* argument does not represent a valid state for the cell, then the value of -2 is returned. If the cell has no state-dependent static probability annotation for the *state* argument, then the value of -1 is returned.

`-path_sources` *name_list*

Specifies the path sources when getting path-dependent toggle rate and glitch rate on pins. This is used when the library cell pin has path-dependent internal power. The path source or sources specified with this argument must be the

same as those in the internal power characterization. The *-path* option takes a path condition (a list of input pins). Again, the path condition after the *-path* option should be enclosed between " ". If there are multiple input pins in the path condition, then each input pin should be separated by a space, for example "A B".

Note that if the *name_list* argument is given using the *-path_sources* option, then the *state* argument should also be provided using the *-state_condition* option.

Given a design object, the command returns a list comprised of object name, the *-state_condition* option, source pin name, toggle rate value, toggle rate annotation source indicator, glitch rate value, and glitch rate annotation source indicator. If the combination of the *state* and *name_list* arguments does not represent a valid state-dependent-path-dependent (SDPD) condition for the design object, then the value of -2 is returned. If the design object does not have toggle rate or glitch rate annotation for the given SDPD conditions, then the value of -1 is returned.

`-toggle_rate`

Specifies the toggle rate (rise and fall, if applicable) which is reported for the given design object. Given a design object, the command returns the list comprised of object name, simple toggle rate value (without glitches), and annotation source indicator. The unit of returned toggle rate value is the number of toggle counts per target technology library time unit. If the object has no toggle rate value annotation, then this command returns the value -1. If the given object is not found in the design, then the value of -2 is returned.

An annotation source indicator can have the following possible values:

- file: implies that the *-toggle_rate* value was annotated through a SAIF or vcd file
- propagated: implies that the *-toggle_rate* value is propagated using random vectors from annotated design points
- default: implies that the *-toggle_rate* value is default
- UNINITIALIZED: implies that the *-toggle_rate* value is uninitialized
- create_clock: implies that the *-toggle_rate* value was annotated by a *create_clock* command
- set_switching_activity: implies that the *-toggle_rate* value was annotated by a *set_switching_activity* command

- set_case_analysis: implies that the *-toggle_rate* value was annotated by a *set_case_analysis* command
- implied: implies that the activity was implied without random vector propagation from annotated design points

The object name is the full name relative to the current instance.

`-glitch_rate`

Specifies the glitch rate (rise and fall, if applicable) which is reported for the given design object. Given a design object, the command returns the list comprised of object name, glitch rate value, and an annotation source indicator. The unit of the returned glitch rate value is the number of glitch counts per target technology library times unit. If the object has no glitch rate value annotation, then this command returns the value -1. If the given object is not found in the design, then the value of -2 is returned.

`-static_probability`

Specifies the static probability value which is reported for the given design object. Given a design object, the command returns the list comprised of object name, simple static probability and an annotation source indicator. The static probability is defined as the percentage of time the signal is at logic state 1. If the object has no static probability value annotation, then this command returns the value -1. If the given object is not found in the design, then the value of -2 is returned.

If none of the *-toggle_rate*, *-glitch_rate* and *-static_probability* options are set, then the command assumes all three options to be true, and returns the *-toggle_rate*, *-glitch_rate*, and *-static_probability* values.

`-rise`

Use with the *-state_condition* option. This option reports either rise or fall transitions. If the *-state_condition* option and *state* argument are given and neither the *-rise* nor *-fall* option is given, then by default the command assumes both to be true. If the *-rise* or *-fall* option is given along with the *-static_probability* option, then the *-rise* and *-fall* options are ignored as static probability does not depend on rise or fall transitions type.

`-fall`

Use with the *-state_condition* option. This option reports either rise or fall transitions. If the *-state_condition* option and *state* argument are given and neither the *-rise* nor *-fall* option is given, then by default the command assumes both to be true. If the *-rise* or *-fall* option is given along with the *-static_probability* option, then the *-rise* and *-fall* options are ignored as static probability does not depend on rise or fall transitions type.

`-only_related_clock` *`clock_name`*

Only nets and objects that are in the clock domain, *clock_name* argument are included. Objects not in the domain are filtered out. When the *-average_activity* option is used, objects are filtered out before averaging takes place.

`-toggle_limit` *`limit`*

Changes the definition of what is considered to be low activity. This definition is used by the *-exclude* and *-include_only* options, depending on the criteria given to those options. The value of the argument represents the maximum number of toggles considered to be low activity. The default is 1. The number of toggles on a net is defined as the toggle rate times the simulation duration. For vector free power analysis (where no SAIF file or VCD is used) the default simulation duration is 10000 ns. If a SAIF is used, the duration is taken from the SAIF file. If a VCD file is used to annotate activity, then the duration is the duration of the VCD simulation.

The *-toggle_limit* option has no effect unless the *-exclude* or *-include_only* options are used. The *-toggle_limit* option only affects the behavior of these other options.

`-exclude` *`exclusion_group`*

This option filters out objects in the *object_list* argument, so that switching activity is not reported for objects that meet the specified criteria.

When the *-average_activity* option is used, the *-exclude* option serves to filter out nets before the average is taken.

For more information about how to use the *-exclude* and *-include_only* options, read the appropriate section in the man page for the *report_switching_activity* command.

`-include_only` *`inclusion_group`*

This option filters out objects in the *object_list* argument, so that switching activity is reported only on objects that meet the specified criteria.

When the *-average_activity* option is used, the *-include_only* option serves to filter out nets before the average is taken.

For more information about how to use the *-exclude* and *-include_only* options, read the appropriate section in the man page for the *report_switching_activity* command.

`-average_activity`

When this option is used, the *-static_probability*, *-rise*, *-fall*, and *-path_sources* options cannot be used.

When the *-average_activity* option is used, the *object_list* argument should consist only of hierarchical cells. For each hierarchical cell given, the average toggle rate and glitch rate of nets within the hierarchical cell is computed and reported.

In this case, the *-include_only* and *-exclude* options play the role of filtering out nets in the hierarchical cell before the average toggle rate is computed.

Static probabilities are never averaged. If the *-average_activity* option is used, only average toggle rate and average glitch rate is reported. Since the nets in the hierarchical cell might have different sources for the activity information, the activity source for the toggle and glitch rate reports as "averaged".

`object_list`

Specifies a list of nets, pins, ports, or cells in the current design for which the *-static_probability*, *-toggle_rate*, and *-glitch_rate* options switching activity values are reported.

When the *-average_activity* option is used, the *object_list* argument should consist of hierarchical cells whose average switching activity on nets in the cell is computed.

`-relative_toggle_rate`

When this option is used, the toggle rate specifies the relative toggle rate with respect to the related clock for the given design object.

Description

This command is used to report different kinds of switching activity information on design nets, ports, pins, and cells. These include simple toggle rate, glitch rate, and static probability on nets, ports, and pins; state and path dependent toggle rate and glitch rate on cell pins, and state dependent static probabilities on cells.

Toggle rates, glitch rates, and static probabilities are reported using the *-toggle_rate*, *-glitch_rate* and *-static_probability* options, respectively. The toggle rate, glitch rate, and static probability can be made state dependent by specifying the state condition with the *-state_condition* option. The toggle rate and glitch rate can be made path dependent by specifying the path source or sources with the *-path_sources* option.

The design objects that are annotated with switching activity can only be specified explicitly as a list of objects. Note that the *object_list* argument should always be provided to the command.

Average activity statistics for a hierarchical block, or portions of a hierarchical block is obtained using the *-average_activity* option.

For more statistics on the switching activity annotation on the current design, use the *report_switching_activity* command.

If the *power_analysis_mode* variable is set to *averaged* or *leakage_variation*, and if a VCD file is read with the *read_vcd* command (without the *-pipe* option), then the tool annotates the activity from the VCD before reporting the switching activity. In addition, unannotated objects where the activity can be derived accurately without random vector simulation (for example, clock nets, Qbar pins where the Q pin is annotated, constant nets) is reported by the *report_switching_activity* command as having the *implied* argument activity. Activity propagation for other unannotated nodes does not happen until the *update_power* command runs.

If the *power_analysis_mode* variable is set to *time_based*, then the *get_switching_activity* command does not have access to toggle rate information before running the *update_power* command. The command reports whether a net is uninitialized or whether it annotates from the VCD file. But since the VCD file has not been read yet, the toggle rate (and other values) is reported as -1.

Use of the *-exclude* and *-include_only* filtering options allow you to filter out objects before reporting switching activity. This is useful, for instance, to report only nets with no switching activity. In this case, the *object_list* argument could be all nets in the design (obtained by the *[get_net * -hierarchical -top]* argument), while the *-include_only {no_switching_activity}* option filters the large set to return a list of only those nets with no switching activity.

Examples

The following *get_switching_activity* command reports simple toggle rate value, glitch rate value, and static probability value on all design input ports.

```
pt_shell> get_switching_activity -toggle_rate -glitch_rate \\
         -static_probability [all_inputs]
pt_shell> get_switching_activity [all_inputs]
```

The following example reports different switching activity values on design output ports.

```
pt_shell> get_switching_activity -toggle_rate \\
          [all_outputs]

pt_shell> get_switching_activity -glitch_rate \\
          [all_outputs]

pt_shell> get_switching_activity -static_probability \\
          [all_outputs]

pt_shell> get_switching_activity -toggle_rate \\
          -static_probability [all_outputs]

pt_shell> get_switching_activity -glitch_rate \\
          -static_probability [all_outputs]

pt_shell> get_switching_activity -glitch_rate \\
          -toggle_rate [all_outputs]
```

```
pt_shell> get_switching_activity -glitch_rate \\
          -toggle_rate -static_probability [all_outputs]
```

The following example reports state dependent static probabilities on the cell or1:

```
pt_shell> get_switching_activity -static_probability \\
          -state_condition "A & B" [get_cell or1]

pt_shell> get_switching_activity -state_condition "A & B" \\
          [get_cell or1]

pt_shell> get_switching_activity -static_probability \\
          -state_condition "A & ! B" [get_cell or1]

pt_shell> get_switching_activity -static_probability \\
          -state_condition "! A & B" [get_cell or1]

pt_shell> get_switching_activity -static_probability \\
          -state_condition "! A & ! B" [get_cell or1]

pt_shell> get_switching_activity -static_probability \\
          -state_condition "default" [get_cell or1]
```

The following example reports simple and path dependent toggle rates and glitch rates on the output pin Y of the cell xor1:

```
pt_shell> get_switching_activity -toggle_rate -glitch_rate \\
          [get_pin xor1/Y]

pt_shell> get_switching_activity -toggle_rate -glitch_rate \\
          -path_sources "A" -state_condition "B" [get_pin xor1/Y]

pt_shell> get_switching_activity -rise -fall -toggle_rate \\
          -glitch_rate -path_sources "A" -state_condition "B" [get_pin
 xor1/Y]

pt_shell> get_switching_activity -rise -toggle_rate \\
          -glitch_rate -path_sources "A" -state_condition "B" [get_pin
 xor1/Y]

pt_shell> get_switching_activity -fall -toggle_rate \\
          -glitch_rate -path_sources "A" -state_condition "B" [get_pin
 xor1/Y]

pt_shell> get_switching_activity -toggle_rate -glitch_rate \\
          -path_sources "B" -state_condition "A" [get_pin xor1/Y]

pt_shell> get_switching_activity -toggle_rate -glitch_rate \\
          -path_sources "A B" -state_condition "default" [get_pin xor1/Y]
```

The following example reports the average switching activity on certain nets within the hierarchical cell "state_machine1". The nets used are those that are driven by sequential cells and have annotated switching activity.

```
pt_shell> get_switching_activity -toggle_rate -average_activity \\
          -include_only {sequential & annotated} [get_cell
 state_machine1]
```

The following example reports the switching activity on every pin that is an output of a sequential cell. This might be useful for checking to see that switching activity annotation to sequential cells happened properly when you invoke a command, such as the *read_saif* command.

```
pt_shell> get_switching_activity  \\
          -include_only {sequential} [get_pin * -hierarchical]
```

The following example reports the switching activity on every pin that is an output of a sequential cell or a black box and does not have annotated activity. This could be useful for generating a list of pins that you might want to annotate before propagation of switching activity.

```
pt_shell> get_switching_activity  \\
          -include_only {sequential,black_box & !annotated} [get_pin *
 -hierarchical]
```

The following example reports the switching activity on every net that has fewer than 10 toggles during simulation. Note that the *get_switching_activity* command reports toggle rates, not toggle counts. To obtain the toggle count of a net, multiply the rate by the simulation time.

```
pt_shell> get_switching_activity  -toggle_limit 10 \\
          -include_only {low_activity} [get_net * -hierarchical -top]
```

The following example reports on a net in the *averaged* power analysis mode. The net is in the VCD file, and the *read_vcd* command had previously been given. This example runs the *get_switching_activity* command before and after the *update_power* argument. The example shows that before power analysis, on the first invocation of an activity reporting command, the VCD activity is applied, then the toggle rate and source are reported. After power analysis, the toggle rate is also reported.

```
pt_shell> get_switching_activity "z[31]"
Information: Reading file vcd.dump.gz to annotate toggle rates on the
 design...

****************************************************************
Summary:
Total number of nets = 1629
Number of annotated nets = 1629 (100.00%)
Total number of leaf cells = 1339
Number of fully annotated leaf cells = 1339 (100.00%)
```

```
=================================================================

 {"z[31]" 0.00330065 file 0 file 0.525616 file}
pt_shell> get_switching_activity "z[31]"
 {"z[31]" 0.00330065 file 0 file 0.525616 file}
pt_shell> update_power
Information: Running average power analysis...
1
pt_shell> get_switching_activity "z[31]"
 {"z[31]" 0.00330065 file 0 file 0.525616 file}
```

The following example reports on a net in the *time_based* power analysis mode. The net is in the VCD file. This example runs the *get_switching_activity* command before and after the *update_power* argument. The example shows that before power analysis, the activity source is known (file), but the toggle rate is unknown and is reported as -1. After the *update_power* argument, the toggle rate is known and is reported.

```
pt_shell> get_switching_activity  "z[31]"
 {"z[31]" -1 file -1 file -1 file}
pt_shell> update_power
Information: The waveform options are:
               File name:      primetime_px.fsdb
               File format:    fsdb
               Time interval:  0.01ns
               Hierarchical level:     1

Information: Power analysis is running, please wait ...

Information: analysis is done for time window (0ns - 9998.03ns)

1
pt_shell> get_switching_activity  "z[31]"
 {"z[31]" 0.00330065 file 0 file 0.526482 file}
```

See Also

- read_saif
- report_power
- report_switching_activity
- reset_switching_activity
- set_switching_activity

---

get_timing_arcs

Creates a collection of timing arcs for custom reporting and other processing. You can assign these timing arcs to a variable and get the desired attribute for further processing.

Feedback

Syntax

string *get_timing_arcs*

```
[-to to_list]
[-from from_list]
[-of_objects object_list]
[-filter expression]
[-quiet]
```

Data Types

to_list	list
from_list	list
object_list	list
expression	string

Arguments

`-to to_list`

Specifies a list of to pins or to ports to get timing arcs for. All backward arcs from the specified pins or ports are considered.

`-from from_list`

Specifies a list of from pins or from ports to get timing arcs for. All forward arcs from the specified pins or ports are considered.

`-of_objects object_list`

Specifies cells or nets to get timing arcs for. If a cell is specified, all cell_arcs of that cell are considered. If a net is specified, all net_arcs of that net are considered.

`-filter expression`

Specifies a string that comprises a series of logical expressions describing a set of constraints you want to place on the collection of arcs. Each subexpression of a filter expression is a relation contrasting an attribute name with a value by means of an operator.

`-quiet`

Specifies that all messages are to be suppressed.

Description

The *get_timing_arcs* command creates a collection of arcs for custom reporting or other operations. Use the *foreach_in_collection* command to iterate among the arcs in the collection. You can use the *get_attribute* command to obtain information about the arcs.

Feedback

You can also use the filter expression to filter the arcs that satisfy the specified conditions. The following attributes are supported on timing arcs:

```
delay_max_fall
delay_max_rise
delay_min_fall
delay_min_rise
from_pin
is_annotated_fall_max
is_annotated_fall_min
is_annotated_rise_max
is_annotated_rise_min
is_cellarc
is_disabled
is_user_disabled
mode
object_class
sdf_cond
sense
to_pin
```

One attribute of a timing arc is from_pin which is the pin or port from which the timing arc begins. In the same way, the to_pin is the pin or port at which the timing arc ends. In order to get more information about the from_pin and the to_pin, use the *get_attributes* command. The *object_class* attribute holds the name of the timing arc class *timing_arc*.

Examples

The following procedure gets the same arguments as the *get_timing_arcs* command and prints out some of the attributes of the timing arcs.

```
proc format_float {number {format_str "%.2f"}} {

  switch -exact -- $number {
    UNINIT { }
    INFINITY { }
    default {set number [format $format_str $number]}
  }
  return $number;
}


proc show_arcs {args} {

  set arcs [eval [concat get_timing_arcs $args]]
  echo [format "%15s    %-15s  %8s %8s  %s" "from_pin" "to_pin" \\
            "rise" "fall" "is_cellarc"]
  echo [format "%15s    %-15s  %8s %8s  %s" "--------" "------" \\
            "----" "----" "----------"]

  foreach_in_collection arc $arcs {
```

Feedback

```
    set is_cellarc [get_attribute $arc is_cellarc]
    set fpin [get_attribute $arc from_pin]
    set tpin [get_attribute $arc to_pin]
    set rise [get_attribute $arc delay_max_rise]
    set fall [get_attribute $arc delay_max_fall]

    set from_pin_name [get_attribute $fpin full_name]

    set to_pin_name [get_attribute $tpin full_name]
    echo [format "%15s -> %-15s  %8s %8s  %s" \\
        $from_pin_name $to_pin_name \\
        [format_float $rise] [format_float $fall] \\
        $is_cellarc]
  }
}
```

```
pt_shell> show_arcs -of_objects ffa

       from_pin    to_pin              rise     fall  is_cellarc
       --------    ------              ----     ----  ----------
         ffa/CP -> ffa/D               0.85     0.85  true
         ffa/CP -> ffa/D               0.40     0.40  true
         ffa/CP -> ffa/Q               1.34     1.42  true
         ffa/CP -> ffa/QN              2.38     1.98  true
         ffa/CD -> ffa/Q               1.00     0.82  true
         ffa/CD -> ffa/QN              1.78     1.00  true

pt_shell> show_arcs -from ffa/CP

       from_pin    to_pin              rise     fall  is_cellarc
       --------    ------              ----     ----  ----------
         ffa/CP -> ffa/D               0.85     0.85  true
         ffa/CP -> ffa/D               0.40     0.40  true
         ffa/CP -> ffa/Q               1.34     1.42  true
         ffa/CP -> ffa/QN              2.38     1.98  true

pt_shell> show_arcs -from ffa/Q

       from_pin    to_pin              rise     fall  is_cellarc
       --------    ------              ----     ----  ----------
          ffa/Q -> QA/A                0.00     0.00  false

pt_shell> show_arcs -to ffa/*

       from_pin    to_pin              rise     fall  is_cellarc
       --------    ------              ----     ----  ----------
         ffa/CP -> ffa/D               0.85     0.85  true
         ffa/CP -> ffa/D               0.40     0.40  true
            a/Z -> ffa/D               0.00     0.00  false
            CLK -> ffa/CP              0.00     0.00  false
          RESET -> ffa/CD              0.00     0.00  false
```

```
ffa/CP -> ffa/Q                1.34     1.42  true
ffa/CD -> ffa/Q                1.00     0.82  true
ffa/CP -> ffa/QN               2.38     1.98  true
ffa/CD -> ffa/QN               1.78     1.00  true
```

See Also

- collections
- foreach_in_collection
- get_attribute

---

get_timing_paths

Creates a collection of timing paths for custom reporting or other processing.

Syntax

collection *get_timing_paths*

```
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-exclude exclude_list
   | -rise_exclude rise_exclude_list
   | -fall_exclude fall_exclude_list]
[-delay_type delay_type]
[-nworst paths_per_endpoint]
[-max_paths max_path_count]
[-group path_group_list]
[-unique_pins]
[-slack_greater_than minimum_slack]
[-slack_lesser_than  maximum_slack]
[-ignore_register_feedback feedback_slack_cutoff]
[-include_hierarchical_pins]
[-trace_latch_borrow]
[-trace_latch_forward]
[-pba_mode none | path | exhaustive | ml_exhaustive]
[-start_end_type from_to_type]
[-normalized_slack]
[-start_end_pair]
[-cover_design]
[-cover_through through_list]
[-dont_merge_duplicates]
[-pre_commands pre_command_string]
```

```
[-post_commands post_command_string]
[-path_type format]
[-attributes attribute_list]
[-tag_paths_filtered_by_pba tag_name]
[-sms_scenarios sms_scenarios_list]
[-domain_crossing domain_crossing_mode]
[-pocv_pruning]
[-vdd_slack_lesser_than maximum_vdd_slack]
[-vdd_slack_greater_than minimum_vdd_slack]
[path_collection]
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
to_list	list
rise_to_list	list
fall_to_list	list
through_list	list
rise_through_list	list
fall_through_list	list
exclude_list	list
rise_exclude_list	list
fall_exclude_list	list
delay_type	string
paths_per_endpoint	integer
max_path_count	integer
path_group_list	list
minimum_slack	float
maximum_slack	float
maximum_vdd_slack	float
minimum_vdd_slack	float
feedback_slack_cutoff	float
from_to_type	string
pre_command_string	string
post_command_string	string
format	string
attribute_list	list
tag_name	string
path_collection	collection
sms_scenarios_list	collection
domain_crossing_mode	string

Arguments

`-from` *from_list*

Collects only paths that start from the specified list of pins, ports, nets, or cell instances; or from startpoints clocked by the named clocks. Valid startpoints are clock pins of sequential devices and input ports of the design.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the path must start with a rising transition. If you specify a clock object, the option selects startpoints clocked by the named clock and launched by a rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_from` *`fall_from_list`*

Same as the *-rise_from* option, except that the path must start with a falling transition.

`-to` *`to_list`*

Collects only paths that end at the specified list of pins, ports, nets, or cell instances; or at endpoints clocked by the named clocks. Valid endpoints are data input pins of sequential devices and output ports of the design.

Due to the infrastructure of the timing analyzer, the *-to* option runs more efficiently than the *-from* option. As a result, endpoint-oriented path gathering (based on using the *-to* option) outperforms equivalent startpoint-oriented path gathering (based on using the *-from* option).

`-rise_to` *`rise_to_list`*

Same as the *-to* option, except that the path data must end with a rising transition. If you specify a clock object, the option selects endpoints clocked by the named clock and captured by a rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_to` *`fall_to_list`*

Same as the *-rise_to* option, except that the path data must end with a falling transition.

`-through` *`through_list`*

Collects only paths in which the data goes through the named pins, ports, cell instances, or nets. You can use this option multiple times in a command.

When a single *-through* list contains multiple objects, the data path can go through any one of the listed objects. When you use multiple *-through* options, the data path must meet each of the *-through* options in sequence.

For example, the following command gets a path that starts at A1, then passes through either B1 or B2, then passes through C1, and ends at D1.

```
pt_shell> report_timing -from A1 -through {B1 B2} -through C1 -to
 D1
```

If advanced analysis through transparent latches is enabled (*timing_enable_through_paths* set to *true*), and the pins specified in the last

-through option are latch loop breakers, then the timing paths collected are constrained by the worst of the closing edges at the latch or by a required value computed using logic downstream of the latch loop breaker.

`-rise_through` *`rise_through_list`*

Same as the *-through* option, except that it applies only to paths with a rising transition at the specified objects.

`-fall_through` *`fall_through_list`*

Same as the *-through* option, except that it applies only to paths with a falling transition at the specified objects.

`-exclude` *`exclude_list`*

Excludes all paths in which the data goes from, through, or to the specified list of pins, ports, nets, or cell instances; or excludes all paths belonging to the specified list of path tag sets. This option checks only data paths (not clock paths) for excluded objects.

Using the *-exclude* option can result in significantly longer runtimes. To reduce the runtime impact, use this option together with other topological restriction options, such as *-from* and *-through*.

The *-exclude* option does not check borrowing paths considered by the *-trace_latch_borrow* option.

`-rise_exclude` *`rise_exclude_list`*

Same as the *-exclude* option, but applies only to rising transitions at the named pins, ports, nets, or cell instances.

`-fall_exclude` *`fall_exclude_list`*

Same as the *-exclude* option, but applies only to falling transitions at the named pins, ports, nets, or cell instances.

You can use no more than one of the *-exclude*, *-rise_exclude*, or *-fall_exclude* options in a command.

`-delay_type` *`delay_type`*

Specifies the type of path delay constraint to consider for finding and sorting paths with the worst slack:

- *max* (default) -- max delay (setup constraint)
- *min* -- min delay (hold constraint)
- *min_max* -- both min and max delay (stores min path first, max second)
- *max_rise* -- max delay, rising at data path endpoint

- *max_fall* -- max delay, falling at data path endpoint
- *min_rise* -- min delay, rising at data path endpoint
- *min_fall* -- min delay, falling at data path endpoint

`-nworst` *`paths_per_endpoint`*

Collects up to the specified number of worst paths per endpoint. The default is 1. A larger value results in a larger collection and more runtime. Allowed values are 1 to 2000000.

If you set *-nworst* to a value greater than 1, the command automatically does the following:

- Implicitly sets *-max_paths* equal to the *-nworst* setting if the *-max_paths* option is not used in the command, so that at least one set of multiple paths to an endpoint can be collected.
- Implicitly sets *-slack_lesser_than 0.0* if the *-slack_lesser_than* and *-slack_greater_than* options are not specified, so that only violators are collected.

`-max_paths` *`max_path_count`*

Collects up to the specified maximum total number of paths.

If the *-group* option is specified, the total number of paths collected is at most the specified number of paths per path group.

If the *-group* option is not specified, the total number of paths collected is at most the specified number of paths among all path groups.

The *max_path_count* must be greater than zero. The default *max_path_count* is equal to the *-nworst* setting, or 1 if the *-nworst* option is not specified.

If you set *-max_paths* to a value greater than 1, the command implicitly sets *-slack_lesser_than 0.0* if the *-slack_lesser_than* and *-slack_greater_than* options are not specified, so that only violators are collected.

The *-nworst* option controls the maximum number of paths collected *per endpoint*, whereas the *-max_paths* option controls the *overall total maximum number* of paths collected. The *-max_paths* setting should be at least as large as the *-nworst* setting, so by default the tool implicitly sets *-max_paths* to the same value as *-nworst* if the *-max_paths* option is not specified.

`-group` *`path_group_list`*

Collects only paths that belong to the specified path groups.

Path groups are created by the *create_clock* and *group_path* commands. Without the *-group* option, the *get_timing_paths* command collects paths without

considering path groups. For example, the *get_timing_paths* command alone creates a collection containing the single worst path in the design, whereas *get_timing_paths -group [get_path_groups]* creates a collection containing multiple paths, consisting of the single worst path *in each path group*.

`-unique_pins`

Collects only the single worst timing path through any given sequence of pins. No other paths are collected for the same sequence of pins from startpoint to endpoint. For example, if the worst path starts with a rising edge at the first pin of a pin sequence, then paths starting with a falling edge are not collected for that sequence of pins.

For non-unate logic such as XOR gates, using this option greatly reduces the number of paths collected because of the large number of possible rising/falling edge combinations through each sequence of pins.

Using this option can require longer runtimes when used with the *-nworst* option because many paths must be analyzed to find the worst path through each pin sequence, but only the worst path is collected and counted toward the total number of requested paths.

`-slack_greater_than` *`minimum_slack`*

Collects only paths with slack greater than the specified *minimum_slack* value; these paths have a negative slack better than the specified *minimum_slack* value (or a positive slack that is farther from causing a violation). This option is intended to be used with the *-slack_lesser_than* option to collect paths within a specific range of slack.

Unlike the *-slack_lesser_than* option, the *-slack_greater_than* option acts in a post-processing way as a filter to restrict the paths that are collected. As a result, the number of paths collected might be less than the number specified with the *-nworst* and *-max_paths* options.

`-slack_lesser_than` *`maximum_slack`*

Collects only paths with slack less than the specified *maximum_slack* value; these paths have a negative slack worse than the specified *maximum_slack* value (or a positive slack that is closer to causing a violation).

If this option is not specified, the default is computed as follows, highest precedence first:

- If exhaustive PBA is used (*-pba_mode* set to *exhaustive* or *ml_exhaustive*), then the default is 0.0, so that only violators are collected.
- Otherwise, if either *-max_paths* or *-nworst* is set to a value greater than 1, then the default is 0.0, so that only violators are collected.

- Otherwise, the default is "infinity", so that the worst path is collected.

The *-slack_lesser_than* and *-slack_greater_than* options act as filters to restrict the paths that are collected, so the number of paths collected can be fewer than the number specified with the *-nworst* and *-max_paths* options. If no paths meet the slack criteria, no paths are collected.

`-ignore_register_feedback` *`feedback_slack_cutoff`*

Ignores noninverting timing loops that start and end at the same register pin that holds a value. To be ignored, the data-to-output arc and the output-to-data path must be either both inverting or both noninverting. This option applies to minimum delay as well as maximum delay constraints. Paths are ignored only if they have a slack less than the specified *feedback_slack_cutoff* value.

This option acts as a filter to restrict the paths that are collected, so the number of paths collected might be less than the number specified with the *-nworst* and *-max_paths* options.

`-include_hierarchical_pins`

Causes the collected paths to include points for the hierarchical pins as well as leaf pins. The hierarchical pins are included for informational purposes only; the *report_timing* command reports them as having zero incremental delay and they do not affect the timing results.

`-trace_latch_borrow`

Controls the type of data collected for a path that starts at a transparent latch.

If the path startpoint borrows from the previous path segment, using this option causes the path data to include the set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. Each path segment is maintained separately, including the time borrowed and lent and the endpoints of the path segment.

Without the *-trace_latch_borrow* option, the command does not include upstream borrowing latches when in default latch analysis mode or borrowing latch loop breakers when in advanced latch analysis mode (*timing_enable_through_paths* set to *true*) as path segments in the timing report.

`-trace_latch_forward`

Controls the type of data collected for a path that ends at a transparent latch D pin.

If advanced analysis through transparent latches is enabled (*timing_enable_through_paths* set to *true*) and this transparent latch D pin is constrained by a downstream required time, this option causes the data to include the paths in the fanout of this D pin that led to the downstream required

constraint. These paths can include multiple path segments whereby the constraint is the result of propagating through more than one latch constrained by the downstream required time. In this case, each path segment is maintained separately, showing the downstream required time for that path segment.

Without the *-trace_latch_forward* option, the command does not trace the path beyond the specified endpoint.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Specifies one of the following path-based timing analysis modes:

- *none* (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- *path* - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths. To control the ordering of the paths (either by original graph-based slack or recalculated path-based slack), set the *pba_path_mode_sort_by_gba_slack* variable.
- *exhaustive* - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode. You cannot use the *exhaustive* mode together with the *-start_end_pair*, *-cover_design*, or *path_collection* options.
- *ml_exhaustive* - Performs Machine Learning based exhaustive path-based analysis. This mechanism deploys machine learning techniques to trade runtime versus accuracy during the analysis. Accuracy and runtime will approach *-pba_mode exhaustive* as the design approaches zero-slack signoff.

In ordinary graph-based (default) timing analysis, the tool considers both the worst arrival time and worst slew among all signals feeding into a path, even when the worst arrival and worst slew come from different signals.

In path-based timing analysis, the tool considers each path in isolation from other paths, which eliminates impossible combinations of worst slew and worst arrival, and similar combinations of effects such as crosstalk and CRPR. As a result, path-based analysis reduces pessimism and increases accuracy at the cost of more runtime.

`-start_end_type` *`from_to_type`*

Restricts the collection to one of four classes of paths based on the type of startpoint and endpoint:

- *reg_to_reg* - from register to register
- *reg_to_out* - from register to output port

- *in_to_reg* - from input port to register
- *in_to_out* - from input port to output port

In this description, "register" means any valid startpoint or endpoint that is not an input port, output port, or bidirectional port, even if not a sequential register. For example, a clock-gating check endpoint qualifies as a "register" in the *in_to_reg* from-to type. A bidirectional port qualifies as both an input port and an output port.

`-normalized_slack`

Collects and sorts paths using normalized slack instead of absolute slack. Normalized slack is the absolute slack divided by an idealized maximum allowable propagation delay (typically the clock period). To use this option, the *timing_enable_normalized_slack* variable must be set to *true* for the current timing update.

`-start_end_pair`

Collects the worst path for each startpoint-endpoint pair in the design.

If *-slack_lesser_than* is not specified, all violating startpoint-endpoint pairs are collected (an implicit *-slack_lesser_than* value of 0.0). If *-slack_lesser_than* is specified, the violating startpoint-endpoint pairs meeting that requirement are collected. The number of paths collected is limited to 2000000 unless *-max_paths* is used to exceed this.

The following options can be used to control the *-start_end_pair* behavior:

```
-max_paths
-nworst
-slack_lesser_than
-from / -rise_from / -fall_from
-through / -rise_through / -fall_through
-to / -rise_to / -fall_to
-exclude / -rise_exclude / -fall_exclude
```

The *-start_end_pair* option can result in very large collections and long runtimes. Use the preceding options thoughtfully to generate the collection. For example, you could use the *-slack_lesser_than* option to specify a slack limit that is only slightly greater than the worst slack in the design.

The following options cannot be used with the *-start_end_pair* option:

```
-cover_design
-slack_greater_than
-unique_pins
-ignore_register_feedback / -report_ignored_register_feedback
-pba_mode exhaustive | ml_exhaustive
path_collection
```

The *-start_end_pair* option sorts the paths first by endpoint and then by slack, with the paths sharing the same endpoint collected together in order of increasing slack.

`-cover_design`

Collects the worst path through each violating pin in the design, so that the collected paths cover every violating pin in the design. A pin is deemed to be violating if its slack is less than zero, or less than the value specified with the *-slack_lesser_than* option if that option is used. The slack of a pin is the worst slack among all paths that start at, pass through, or end at that pin. There is no limit on the number of returned paths.

The following options can be used to control the *-cover_design* behavior:

-slack_lesser_than
-from / -rise_from / -fall_from
-through / -rise_through / -fall_through
-to / -rise_to / -fall_to
-exclude / -rise_exclude / -fall_exclude

The following options cannot be used with the *-cover_design* option:

-nworst
-max_paths
-start_end_pair
-unique_pins
-ignore_register_feedback
-pba_mode exhaustive | ml_exhaustive
path_collection

`-cover_through` *through_list*

Collects the single worst violating path through each of the named pins, ports, cell instances, and nets. Only paths with negative slack are collected (or with slack less than the value specified by the *-slack_lesser_than* option, if used).

The number of paths collected is less than or equal to the number of objects in the *through_list*. Fewer paths are collected if there are no violating paths through the objects, or if the worst path through one object is the same as the worst path through another object; duplicate paths are combined into a single path.

The following options cannot be used with the *-cover_through* option:

-nworst
-max_paths
-start_end_pair
-unique_pins
-through / -rise_through / -fall_through
-exclude / -rise_exclude / -fall_exclude
-ignore_register_feedback

-pba_mode exhaustive | ml_exhaustive
path_collection

-dont_merge_duplicates

Prevents the merging of duplicate paths across multiple scenarios in distributed multi-scenario analysis (DMSA).

By default, when the same path is collected in more than one scenario, the tool keeps only the single most critical instance of that path in the merged report and shows its associated scenario.

If you use this option, the tool does not merge duplicate instances of the same path, but instead collects all critical instances of the path from all scenarios. Because the number of paths collected might be limited by *-nworst*, *-max_paths*, or other command options, the resulting merged collection might be more focused on a portion of the design that is critical in multiple scenarios, and less evenly spread out across different portions of design.

-pre_commands *pre_command_string*

Specifies a list of commands, separated by semicolons, to be executed in the worker context before execution of the *get_timing_paths* command. This option is available only if you invoke the PrimeTime tool with the *-multi_scenario* option. The maximum size of a command is 1000 characters.

-post_commands *post_command_string*

This option is like the *-pre_commands* option, except that the commands are executed after (instead of before) the *get_timing_paths* command.

-path_type *format*

Specifies whether propagated clock path information should be included in the collected timing path objects.

Use the *-path_type full_clock_expanded* option to include clock path information, stored in *launch_clock_paths* and *capture_clock_paths* attributes of each timing path object. These attributes include one or more *timing_path* objects representing the propagated clock path segments, including segments for the source latency paths of any generated clocks.

You can query these clock path attributes to obtain information about them. If the path object is later reported with the *report_timing* command, the clock paths are included in the timing report.

By default (when this option is omitted), propagated clock path information is not included in the timing path objects.

Note that while the *report_timing* command supports several *-path_type* values, the *get_timing_paths* command supports only *full_clock_expanded*. If you

specify a value of *full* or *full_clock*, the *get_timing_paths* command interprets it as *full_clock_expanded*.

`-attributes` *`attribute_list`*

Specifies the attributes to be returned to the distributed manager from a worker path collection.

By default, a worker collection returns only the *full_name*, *scenario_name*, and *object_class* attributes. To control how much data is retrieved from the worker, use this option together with the *set_distributed_parameters -collection_levels* command.

This option is available only in distributed multi-scenario analysis (DMSA).

`-tag_paths_filtered_by_pba` *`tag_name`*

When used with the *-pba_mode path* option, the *-tag_paths_filtered_by_pba* option causes tagging of all paths originally found by path-based analysis and later discarded by path-based analysis, using the specified tag name. In future *get_timing_paths* and *report_timing* commands, you can exclude the tagged paths from further analysis by specifying the tag name as an argument to the *-exclude* option.

`path_collection`

Performs path-based analysis on the specified paths and returns a new path collection. You can use this argument only with the *-pba_mode path* option. This option is mutually exclusive with options that control the selection of paths to collect.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only paths compatible to the specified SMS scenarios collection will be returned. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

`-domain_crossing` *`domain_crossing_mode`*

Specifies the domain crossing report mode. Allowed values include *all, only_crossing, exclude_crossing*. The default mode is *all*. This option is applied as a reporting filter, ensuring that only paths which satisfy the specified mode are reported. With the default setting of *all*, all paths are considered for reporting, including cross-domain and intra-domain. With the *only_crossing* mode, reports include only cross-domain paths. Using the *exclude_crossing* mode only reports intra-domain paths.

This option is only available with SMVA/SMC analysis.

`-pocv_pruning`

Restricts the path collection to only those paths that contribute to joint success rate. This option prunes away all subcritical paths that have no impact on the high-sigma failure rate.

This option should always be used when obtaining paths to perform design variation analysis with the *get_design_variation* command.

This option works only with the *-pba_mode exhaustive* option. It can significantly reduce runtime and memory in both the *get_timing_paths* and *get_design_variation* commands.

`-vdd_slack_lesser_than` *`maximum_vdd_slack`*

Performs Vmin analysis on the paths and assigns voltage slack in the path if the value is less than the specified *maximum_vdd_slack value.*

The *-vdd_slack_lesser_than* option must be used with *-pba_mode path*| *exhaustive*.

`-vdd_slack_greater_than` *`minimum_vdd_slack`*

Performs Vmin analysis on the paths and assigns voltage slack in the path if the value is greater than the specified *minimum_vdd_slack value.*

The *-vdd_slack_greater_than* option must be used with *-pba_mode path*| *exhaustive*.

Description

The *get_timing_paths* command creates a collection of paths for custom reporting or other operations. Most of the *get_timing_paths* command options are shared by the *report_timing* command, and the behavior of these shared options is identical. The order in which paths are returned from *get_timing_paths* matches the reported timing path order of *report_timing*.

You can pass the generated collection directly to another command, or you can set a variable to the collection for later reporting. For example,

```
pt_shell> sizeof_collection [get_timing_paths -nworst 4 -max_paths 20]
Warning: report_timing has satisfied the max_paths criteria. ...
20

pt_shell> set mypaths [get_timing_paths -nworst 4 -max_paths 20]
Warning: report_timing has satisfied the max_paths criteria. ...
Information: Defining new variable 'mypaths'. (CMD-041)
 ...
pt_shell> sizeof_collection $mypaths
20
pt_shell> get_attribute $mypaths slack
-1.157432 -0.987188 -0.984205 -0.971914 -0.968931 -0.873521 ...
```

```
pt_shell> report_timing $mypaths
(20 path timing reports displayed)
```

To iterate over the paths in a collection, use the *foreach_in_collection* command. To obtain information about paths, use the *get_attribute* and *report_attribute* commands. To get a list of timing path attributes, use the following command:

```
pt_shell> list_attributes -application -class timing_path
 ...
Attribute Name             Object        Type       Properties
 Constraints
------------------------------------------------------------------------
---
arrival                    timing_path   float      A
capture_clock_paths        timing_path   collection A
clock_uncertainty          timing_path   float      A
close_edge_adjustment      timing_path   float      A
common_path_pessimism      timing_path   float      A
 ...
```

For more information on these *timing_path* attributes, see the *timing_path_attributes* man page.

User-defined attributes can be created for timing paths and set on specific timing paths with the *define_user_attribute* and *set_user_attribute* commands. To be queried, the attribute must be set on a timing path collection that has been stored in a variable. For example,

```
pt_shell> define_user_attribute -classes timing_path -type boolean
 is_special
pt_shell> set mypath [get_timing_paths ...]
pt_shell> set_user_attribute -class timing_path $mypath is_special true
pt_shell> get_attribute $mypath is_special
true
```

This behavior differs from user attributes set on other objects, such as pins, which allow user attributes to be set and queried directly in the design (not stored in a collection variable).

The *get_timing_paths* command constructs a new timing path at every invocation. Timing paths derived from two identical *get_timing_paths* commands and stored in two different collection variables do not share the same user attributes. For example,

```
pt_shell> define_user_attribute comment -type string -class timing_path
pt_shell> set p1 [get_timing_paths]
pt_shell> set p2 [get_timing_paths]
pt_shell> set_user_attribute $p1 comment "Hello world"
pt_shell> get_attribute $p1 comment
Hello world
pt_shell> get_attribute $p2 comment
```

```
Warning: Attribute 'comment' does not exist on timing_path
 'path' (ATTR-3)
```

Each *timing_path* object contains a collection of *timing_point* objects (stored in the *points* attribute of the path). A "point" corresponds to a pin or port along the path. You can iterate through these points with the *foreach_in_collection* command and query their attributes by using the *get_attribute* command. To get a list of timing point attributes, use the following command:

```
pt_shell> list_attributes -application -class timing_point
 ...
Attribute Name              Object         Type     Properties
 Constraints
------------------------------------------------------------------------
---
annotated_delay_delta       timing_point   float    A
annotated_delta_transition  timing_point   float    A
aocvm_coefficient           timing_point   float    A
applied_derate              timing_point   float    A
arrival                     timing_point   float    A
 ...
```

For more information on these *timing_point* attributes, see the *timing_point_attributes* man page.

For information about creating collections and iterating over the elements of a collection, see the examples in the EXAMPLES section and the man pages of the *collections* and *foreach_in_collection* commands.

To report the dominant and overridden timing exceptions related to a timing path, use the *report_timing* command with the *-exceptions* option. To get more information about the exceptions that apply to the path, you can create a path collection with the *get_timing_paths* command and query the following timing path attributes:

```
Attribute                          Possible Reason
-------------------------------------------------------------------
dominant_exception                 false_path / min_max_delay /
                                   multicycle_path / path_margin
startpoint_unconstrained_reason    no_launch_clock /
 dangling_start_point /
                                   fanout_of_disabled
endpoint_unconstrained_reason      no_capture_clock / dangling_end_point /
                                   fanin_of_disabled
```

Examples

You can find some detailed examples of custom timing reports in:

```
$SYNOPSYS_ROOT/auxx/pt/examples/tcl
```

where $SYNOPSYS_ROOT is the installation directory for PrimeTime.

The following procedure reports the startpoint name, endpoint name, and the slack of the worst path in each path group.

```
proc custom_report_worst_path_per_group {} {
  echo [format "%-20s %-20s %7s" "From" "To" "Slack"]
  echo "-------------------------------------------------------------"
  foreach_in_collection path [get_timing_paths] {
    set slack [get_attribute $path slack]
    set startpoint [get_attribute $path startpoint]
    set endpoint [get_attribute $path endpoint]
    echo [format "%-20s %-20s %s" [get_attribute $startpoint full_name]
 \\
      [get_attribute $endpoint full_name] $slack]
  }
}

pt_shell> custom_report_worst_path_per_group

>From                 To                     Slack
-------------------------------------------------------
ffa/CP               QA                   0.1977
ffb/CP               ffd/D                3.8834
```

The following example shows Total Negative Slack, Total Positive Slack, and Worst Negative Slack for the current design.

```
proc report_design_slack_information {} {
  set design_tns 0
  set design_wns 100000
  set design_tps 0
  foreach_in_collection group [get_path_groups *] {
    set group_tns 0
    set group_wns 100000
    set group_tps 0
    foreach_in_collection path [get_timing_paths -nworst 10000 -group
 $group] {
      set slack [get_attribute $path slack]
      if {$slack < $group_wns} {
        set group_wns $slack
        if {$slack < $design_wns} {
           set design_wns $slack
        }
      }
      if {$slack < 0.0} {
        set group_tns [expr $group_tns + $slack]
      } else {
        set group_tps [expr $group_tps + $slack]
      }
    }
    set design_tns [expr $design_tns + $group_tns]
    set design_tps [expr $design_tps + $group_tps]
    set group_name [get_attribute $group full_name]
```

```
    echo [format "Group '%s' Worst Negative Slack : %g" $group_name
 $group_wns]
    echo [format "Group '%s' Total Negative Slack : %g" $group_name
 $group_tns]
    echo [format "Group '%s' Total Positive Slack : %g" $group_name
 $group_tps]
    echo ""
  }
  echo "-----------------------------------------"
  echo [format "Design Worst Negative Slack : %g" $design_wns]
  echo [format "Design Total Negative Slack : %g" $design_tns]
  echo [format "Design Total Positive Slack : %g" $design_tps]
}

pt_shell> report_design_slack_information

Group 'CLK' Worst Negative Slack : -3.1166
Group 'CLK' Total Negative Slack : -232.986
Group 'CLK' Total Positive Slack : 4.5656

Group 'vclk' Worst Negative Slack : -4.0213
Group 'vclk' Total Negative Slack : -46.1982
Group 'vclk' Total Positive Slack : 0

-----------------------------------------
Design Worst Negative Slack : -4.0213
Design Total Negative Slack : -279.184
Design Total Positive Slack : 4.5656
```

If you use the *-pba_mode* option, path recalculation (path-based analysis) is used during the path search, and the worst recalculated paths meeting your criteria are returned. This option requires that a path search to be performed to ensure that the paths being returned truly are the worst paths. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases. Large *-nworst* values can significantly increase the time needed for the search.

To see the worst path in the design with path-based analysis applied:

```
pt_shell> report_timing [get_timing_paths -pba_mode exhaustive]
```

To report all endpoints in the design which fail after considering path-based analysis:

```
pt_shell> set mypaths [get_timing_paths -pba_mode exhaustive \\
            -slack_lesser_than 0 -max_paths 1000]
pt_shell> report_timing $mypaths
```

Feedback

See Also

- collections
- create_clock
- foreach_in_collection
- get_attribute
- group_path
- read_sdf
- report_timing
- set_case_analysis
- set_operating_conditions
- timing_report_always_use_valid_start_end_points

---

get_timing_yield

This is a synonym for the *get_design_variation* command.

Syntax

collection *get_design_variation*

```
path_collection
[-sample_size number]
```

See Also

- get_design_variation
- yield_enable_analysis
- get_timing_paths
- report_timing_yield
- report_yield_bottleneck
- report_cell_robustness
- report_voltage_robustness

Feedback

get_trace_option

Get the current option value controlling behavior of command tracing and output annotation..

Syntax

get_trace_option [-command name |-profile |-memory_threshold |-cpu_threshold |-time_treshold] [-annotate | -is_traced]

string *name*

Arguments

`-command` *name*

This option is mutually exclusive with -profile, -memory_threshold, -cpu_threshold, and -time_threshold. The option identifies the command for which tracing or annotation option is returned.

`-annotate`

If this option is given, then the *annotate* setting for the given command is returned. This option is mutually exclusive with -is_traced and is meaningful only in combination with the -command option.

`-is_traced`

If this option is given, then returns 1 if the given command is traced and logged, 0 otherwise. This option is mutually exclusive with -annotate and is meaningful only in combination with the -command option.

`-profile`

This option is mutually exclusive with all others. The option returns a list of the currently enabled profile metrics, or "all" if all metrics are enabled.

`-memory_threshold`

This option is mutually exclusive with all others. The option returns the currently set memory profile threshold, if any, in kilobytes. If no threshold is set, then it returns the string "unset".

`-cpu_threshold`

This option is mutually exclusive with all others. The option returns the currently set cpu profile threshold, if any, in seconds. If no threshold is set, then it returns the string "unset".

Feedback

`-time_threshold`

This option is mutually exclusive with all others. The option returns the currently set wall clock time profile threshold, if any, in kilobytes. If no threshold is set, then it returns the string "unset".

Description

The command is used to query the current setting for the command annotation option that is set with the set_trace_option command, or profile metric settings.

Examples

The following example sets the annoation type of the *get_attribute* command.

```
prompt> get_trace_option -command get_attribute -annotate
annotate
```

The following example sets, then gets, the currently enabled profile metrics.

```
prompt> set_trace_topion -profile {cpu time}
prompt> get_trace_topion -profile
cpu time
```

The following example gets the threshold for memory profile metric before one has been set.

```
prompt> get_trace_option -memory_threshold
unset
```

The following example will list the commands in the Builtins command group that are not traced and logged.

```
prompt> foreach cmd [dict get [get_defined_commands -details -groups
 Builtins] commands] {
?  if {![get_trace_option -command $cmd -is_traced]}
?    {echo $cmd}
?  }
```

See Also

- log_trace
- annotate_trace
- set_trace_option

---

get_unix_variable

This is a synonym for the *getenv* command.

Feedback

See Also

- printenv
- printvar
- setenv
- sh

---

get_vias

Creates a collection by selecting vias from the design.

Syntax

```
collection get_vias
[-of_objects objects]
```

Data Types

```
objects        collection
```

Arguments

`-of_objects` *of_objects*

Creates a collection containing the vias connected to the specified net. Each object in the list is a collection.

Description

This command creates a collection of vias by selecting vias from the current design that meet the selection criteria. It returns a collection handle if one or more vias meet the selection criteria. If no vias match the selection criteria, it returns an empty string.

Use the *get_vias* command as an argument to another command or assign its result to a variable. Refer to the example below for more information.

See the collection command man page for information about working with collections.

Examples

The following examples create via collections of net n300:

```
prompt> get_vias -of_objects [get_nets n300]
{"VIA_C_67"}
```

Feedback

See Also

- get_attribute
- query_objects

---

get_voltage_areas

Creates a collection of voltage_areas from the current design.

Syntax

```
collection get_voltage_areas
```

Data Types

```
name          string
```

Arguments

```
name design
```

Specifies the net name for finding objects. The net and the voltage areas will be found in the top design

Description

This command creates a collection of voltage_areas from the current design that match certain criteria. The command returns a collection handle (identifier) if any voltage_areas match the value of the *patterns* option and pass the filter (if specified). If no objects match the criteria, the command returns an empty string.

You can use the *get_voltage_areas* command at the command prompt or nest it as an argument to another command.

You can also assign the *get_voltage_areas* result to a variable.

See the *collections* command man page for information about working with collections.

Examples

The following returns voltage_area from its shapes

```
prompt> get_voltage_areas net321
{VA_1}
```

Feedback

See Also

- create_voltage_area
- get_placement_blockages
- get_attribute

---

getenv

Returns the value of a system environment variable.

Syntax

string *getenv*

```
        variable_name
```

Data Types

```
variable_name      string
```

Arguments

variable_name

Specifies the name of the environment variable to be retrieved.

Description

The *getenv* command searches the system environment for the specified *variable_name* and sets the result of the command to the value of the environment variable. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the *env* Tcl array variable. The *getenv*, *setenv*, and *printenv* environment commands are convenience functions to interact with this array.

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you set the variable to a new value using the *setenv* command, you see the new value within the application and within any new child processes you initiate from the application using the *exec* command. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the *exec* command, that value is not reflected in the current application.

See the *set*, *unset*, and *printvar* commands for information about working with non-environment variables.

Feedback

Examples

In the following example, *getenv* returns you to your home directory:

```
prompt> set home [getenv "HOME"]
/users/disk1/bill

prompt> cd $home

prompt> pwd
/users/disk1/bill
```

In the following example, *setenv* changes the value of an environment variable:

```
prompt> getenv PRINTER
laser1

prompt> setenv PRINTER "laser3"
laser3

prompt> getenv PRINTER
laser3
```

In the following example, the requested environment variable is not defined. The error message shows that the Tcl variable *env* was indexed with the value UNDEFINED, which resulted in an error. In the second command, *catch* is used to suppress the message.

```
prompt> getenv "UNDEFINED"
Error: can't read "env(UNDEFINED)": no such element in array
        Use error_info for more info. (CMD-013)

prompt> if {[catch {getenv "UNDEFINED"} msg]} {
         setenv UNDEFINED 1
       }
```

See Also

- printenv
- printvar
- setenv
- unsetenv

---

group_path

Groups paths for reporting.

Feedback

Syntax

status *group_path*

```
-name group_name
-default
[-weight weight_value]
[-from from_list]
[-rise_from rise_from_list]
[-fall_from fall_from_list]
[-to to_list]
[-rise_to rise_to_list]
[-fall_to fall_to_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-comment comment_string]
```

Data Types

group_name	string
weight_value	float
from_list	list
rise_from_list	list
fall_from_list	list
to_list	list
rise_to_list	list
fall_to_list	list
through_list	list
rise_through_list	list
fall_through_list	list
comment_string	string

Arguments

`-name` *group_name*

Specifies a name for the group. If a group with this name already exists, the command adds the paths or endpoints to that group. If a group with this name does not exist, the command creates a new group.

You must use either the *-name* option or the *-default* option.

`-default`

Specifies that endpoints or paths are moved to the default group and removed from the current group.

`-weight` *weight_value*

Specifies a cost function weight for this group, which is used by the Design Compiler tool to prioritize optimization. The *weight_value* value must be a number between 0.0 and 100.0, the default is 1.0. A weight of 0.0 eliminates the

paths in this group from cost function calculations. Do not use very small values (for example, 0.0001).

`-from` *`from_list`*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected.

When the timing path has interior transparent latches, the *-from*, *-rise_from*, and *-fall_from* options refer to the clock, clock pin, or data pin of the last interior latch on the path, not to the initial startpoint of the timing path.

You can use only one of the *-from*, *-rise_from*, and *-fall_from* options.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_from` *`fall_from_list`*

Same as the *-from* option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-to` *`to_list`*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If you specify a cell, one path endpoint on that cell is affected.

You can use only one of the *-to*, *-rise_to*, and *-fall_to* options.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

`-fall_to` *`fall_to_list`*

Same as the *-to* option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-through` *`through_list`*

Specifies a list of path throughpoints (ports, pins, cells, or nets).

You can use multiple *-through*, *-rise_through*, and *-fall_through* options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one *-through*, *-rise_through*, or *-fall_through* option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

`-rise_through` *`rise_through_list`*

It is similar to the *-through* option, but applies only to paths with a rising transition at the specified objects.

`-fall_through` *`fall_through_list`*

It is similar to the *-through* option, but applies only to paths with a falling transition at the specified objects.

`-comment` *`comment_string`*

Associates a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

Description

The *group_path* command groups a set of paths for reporting by the *report_timing* and *report_constraint* commands.

By default, the PrimeTime tool creates a single clock group for each clock created by the *create_clock* command, using the same name for the group as the clock name. By using the *group_path* command to assign paths to specific clock groups, you control the scope of path reporting from different groups.

By default, the *report_timing* command reports the worst setup path in each clock group (or only the clock groups specified by the *-group* option). For example, if you combine all the paths clocked by CLK1 and CLK2 into a new group called CLKN, then the *report_timing* command reports only the single worst path from the combined group CLKN. On the other hand, if you put all paths that end at output OUT1 into a new group called END-OUT1, the *report_timing* command reports the worst path that ends at OUT1, even if other paths clocked by the same clock have worse timing.

If you specify a clock as the "from" object, all paths with startpoint clocked by that clock are included in the group.

If you specify a clock as the "to" object, all paths with an endpoint clocked by that clock are included in the group.

When multiple path group definitions apply to a path, the more specific definition overrides the more general. They apply with the following precedence (more specific to more general):

```
1. group_path -from pin -to pin
2. group_path -from pin -to clock
3. group_path -from pin
4. group_path -from clock -to pin
5. group_path -to pin
6. group_path -from clock -to clock
7. group_path -from clock
8. group_path -to clock
```

If multiple path groups definitions with the same object specification precedence apply to the same path, the path group whose name comes first in alphanumeric order is applied.

The PrimeTime tool creates default groups for clock gating checks and asynchronous preset/clear checks named *"**clock_gating_default**"* and *"**async_default**"* respectively. You can move the paths belonging to these groups into user-defined groups by using the *group_path* command.

Note that the *group_path* command is not compatible with hierarchical model extraction.

To undo the *group_path* command, use the *remove_path_group* command. To report path group information for a design, use the *report_path_group* command.

When a path has transparent latches in the interior of the path, *group_path* is only applied when all the path specifiers are satisfied at or after the last interior latch on the path. The *-from* parameter refers to the clock, clock pin, or data pin of the last interior latch. The *-through* pins must be after the last interior latch.

Examples

The following example groups all endpoints clocked by CLK1A or CLK1B into a new group 'group1'.

Feedback

```
pt_shell> group_path -name group1 -to {CLK1A CLK1B}
```

The following example adds paths that end at OUT1 and ff34/D to the existing path group named 'ADDR'.

```
pt_shell> group_path -name ADDR -to {OUT1 ff34/D}
```

The following example removes the paths that end at OUT1 and CLK2 from existing groups and places them in the default group.

```
pt_shell> group_path -default -to {OUT1 CLK2}
```

The following example groups all paths from inputs I1 and I2 to outputs O5 and O7 into a group named 'serious'.

```
pt_shell> group_path -name serious -from {I1 I2} -to {O5 O7}
```

See Also

- create_clock
- current_design
- remove_path_group
- report_constraint
- report_path_group
- report_timing
- reset_design
- set_input_delay
- set_output_delay

---

gui_add_annotation

Adds an annotation to the layout window.

Syntax

```
status gui_add_annotation
[-window window_name]
[-group group_type]
[-type shape_type]
[-symbol_type symbol_type]
[-symbol_size symbol_size]
[-text text]
[-color color]
[-pattern pattern]
```

Feedback

```
[-width line_width]
[-line_style line_style]
[-visible visible]
[-info_tip info_tip]
[-query_text query_text]
[-query_command query_command]
[-radius radius]
points
```

Data Types

window_name	string
group_type	string
shape_type	string
symbol_type	string
symbol_size	string
text	string
color	string
pattern	string
line_width	string
line_style	string
visible	boolean
info_tip	string
query_text	string
query_command	string
radius	float
points	list

Arguments

`-window` *window_name*

Specifies the instance name of the layout window. If window name is omitted, the annotation will apply to all layout windows.

`-group` *group_type*

Specifies the annotation group. If group name is omitted, the *global* name is used.

`-type` *shape_type*

Specifies the type of annotation. The supported values for this option are

rect
line
arrow
polygon
polyline
text
symbol
ruler
manhattan_ruler

circle
circular_ruler

The default is symbol, rect, or polygon if the number of points is 1, 2, or larger than 2 respectively.

`-symbol_type` *symbol_type*

Specifies the type of symbol annotation. This option is only valid when annotation type is symbol. The supported values for this option are

square
x
triangle
diamond

The default is x.

`-symbol_size` *symbol_size*

Specifies the size of symbol annotation. An even symbol size will grow one automatically to find the appropriate center point. The default symbol size is 25.

`-text` *text*

Specifies the annotation text.

`-color` *color*

Specifies the annotation color. You can use a predefined color string, such as white, red, green, blue, yellow, and so forth, or a hexadecimal RGB value, such as #AA6633. The default annotation color is white. Annotations use colors from the qt color palette by default. Colors from other color palettes may be used by specifying a color name qualified with a palette name. For example, "red:highlight" specifies the red color from the highlight palette. Available palettes are qt, highlight, highcontrast, and style. Use gui_get_color_value command to report on color names and values available in a palette.

`-pattern` *pattern*

Specifies the annotation fill pattern. The default fill pattern is none. An empty value forces the tool to use the window default value, which is usually set to solid fill.

`-width` *line_width*

Specifies the annotation line width. The default line width is 1.

`-line_style` *line_style*

Specifies the annotation line style. The default line style is none. An empty value forces the tool to use the window default value, which is usually set to solid line.

Feedback

`-visible` *visible*

Specifies the visibility of the annotation. The default visible is true.

`-info_tip` *info_tip*

Specifies an info tip for this annotation. When the user starts the query tool in the layout, and puts the mouse cursor over this annotation the specified text will be displayed as the infoTip.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

```
%window  the layout window name
%view    the layout view name
%object  a collection containing the annotation
```

`-query_text` *query_text*

This option is only valid if an info tip was specified. Use this to specify a more detailed string that is displayed in the query palette when the object is selected via the query tool. If query text is not specified then the query tool will just use the info tip string.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

```
%window  the layout window name
%view    the layout view name
%object  a collection containing the annotation
```

`-query_command` *query_command*

This option is only valid if an info tip was specified. Use this to specify a tcl command to be run when the user clicks on the annotation.

The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

```
%window  the layout window name
%view    the layout view name
%object  a collection containing the annotation
%x       the x position clicked in design coordinates
%y       the y position clicked in design coordinates
```

Feedback

`-radius` *radius*

Specifies the radius of circle annotation. This option is only valid when annotation type is circle.

points

Specifies the points for the specified annotation type. The points are specified by a tool command language (Tcl) list in which each list element is one of the following:

- the {x y} location of a point
- a collection containing a single object that has a visual representation in the layout. In other words, it is not a net, for example. The location of the point is the origin of the object.

If a collection is specified the list element can optionally include a position type, one of:

- center : the annotation is placed at the center of the largest rectangle of the object. The largest rectangle is the rectangle with the most area which is completely contained inside the object shape.
- bbox_center : the annotation is placed at the center of the bounding box of the object.
- bbox_ll : the annotation is placed at the lower left of the bounding box of the object.
- bbox_lr : the annotation is placed at the lower right of the bounding box of the object.
- bbox_ul : the annotation is placed at the upper left of the bounding box of the object.
- bbox_ur : the annotation is placed at the upper right of the bounding box of the object.

If a bounding box position type is specified, i.e. one of 'bbox_center', 'bbox_ll', 'bbox_lr', 'bbox_ul' or 'bbox_ur', then the list element can also optionally include an x and y fractional offset from this bounding box point. The x and y offset is a fraction of the width and height of the object's bounding box respectively. For example, for a 'bbox_center' position type, an x offset of -0.5 is the left edge and an x offset of 0.5 is the right edge. Similarly a y offset of -0.5 is the bottom edge and a y offset of 0.5 is the top edge.

For annotations referring to a single-object collection, if the object is moved, the tool updates the point automatically as needed.

Feedback

Description

This command defines a new annotation and returns a collection containing that annotation if it is successful.

Examples

```
Create a green rectangle annotation.

prompt> gui_add_annotation -window Layout.1 \\
   -type rect -color green {{200 200} {777 777}}

Create a light_green rectangle annotation using light_green from the
 highcontrast palette.

prompt> gui_add_annotation -window Layout.1 \\
   -type rect -color light_green:highcontrast {{200 200} {777 777}}

Create a light_green x annotation of size 100 using light_green from the
 highcontrast palette.

prompt> gui_add_annotation -window Layout.1 \\
   -type x -symbol_size 100 -color light_green:highcontrast [list
 {1835.0010 119.2950}]

Create a red polyline annotation in the group Test1.

prompt> gui_add_annotation -window Layout.2 \\
   -group Test1 -type polyline -color red \\
   -width 2 {{123 456} {789 12} {200 200}}

Create a line between two cells A and B.

prompt> gui_add_annotation -window Layout.1 \\
   -type line [list [get_cells A] [get_cells B]]

Create a line between the center of largest rectangle of cell A and the
 center of
the largest rectangle of cell B.

prompt> gui_add_annotation -window Layout.1 \\
   -type line [list [list [get_cells A] center] [list [get_cells B]
 center]]

Create a line between the bottom left of cell A's bounding box and the
 middle of
the top of cell B's bounding box.

prompt> gui_add_annotation -window Layout.1 -type line \
   [list [list [get_cells A] bbox_ll] [list [get_cells B] bbox_center 0.0
 0.5]]
```

Feedback

```
Create a blue rectangle which displays the annotation points using the
 annotation_query
tcl procedure.

prompt> proc annotation_query { object window view } { \\
  return [get_attribute $object points] \\
  }

prompt> gui_add_annotation -window Layout.1 -type rect -color blue \\
  -query_text {=annotation_query %object %window %view} -info_tip
 {annotation points} \\
  {{110 110} {160 160}}

Create a circle annotation whose center is {100 100} and radius is 10.

prompt> gui_add_annotation -type circle -radius 10 {{100 100}}

Create a circle annotation whose center is {100 100} and boundary is at
 point {120 120}

prompt> gui_add_annotation -type circle {{100 100} {120 120}}

Create a circular ruler annotation with 3 circles (radius 20, 40, 100) at
 100, 100.

prompt> gui_add_annotation -type circular_ruler {{100 100} {120 100} {140
 100} {200 100}}
```

See Also

- gui_remove_annotations
- gui_remove_all_annotations
- gui_get_color_value

---

gui_append_utable

Append data rows to an existing UserTable.

Syntax

string *gui_append_utable*

```
-name                  Name
-rows                  TCL_Data_List

-import                File_Name [-val_map ...] [-col_map ...] [-tsv_mode]
 [-ssv_mode]
-val_map               Column_Value_Map
-col_map               Column_Value_Map
```

```
[-tsv_mode]
[-ssv_mode]

-column                Name
[-expr]                Value

Name             String
File_Name        String
Column_Value_Map    TCL List of column value pairs
TCL_DataList     TCL List of String List
Value            Value String
```

Arguments

`-name` *Name*

The name of an existing UserTable that the new data rows should be added to.

`-rows` *TCL_Data_List*

A TCL list of string lists that match the number of columns.

`-import` *File_Name*

A value file that is imported and appended to the given UserTable.

`-val_map` *Column_Value_Map*

A list of column-value pairs that are used to fill empty data values for a given column when importing a file.

`-col_map` *Column_Value_Map*

A list of column-value pairs that are used to map a table column name to the imported table column name. The default is to map to the same name. But if the name of a column in the imported file is different then use this option.

`-tsv_mode`

This flag changes the default delimiter from the comma char to the tab char.

`-ssv_mode`

This flag changes the default delimiter from the comma char to the semicolon char.

Description

```
This command appends new data to and existing table. This can be done by
 either
appending a TCL list of lists as new rows, or by importing CSV files and
 appending
them. Also you have the ability to add a new column that can also be
 filled
```

Feedback

with a math expression that can use +,-,* and / on referenced column values in
the same row.

When appending a TCL list of lists as new rows, row columns are filled left to right,
first to last, in a given list of lists. Extra data values in a row list are ignored
and if the row list is too short on values it will fill the extra columns with
empty fields. If the **-rows** argument is just a single list of values, it is
assumed to be only one row of values that will be added. Again extra row values
are ignored and missing row values are filled with empty fields.

Another way to add new rows to an existing table is to import another table and
append its content to this table. Use the -import option along with related options
-val_map and **-col_map**. Only columns that match the target table are imported
unless you use the -col_map option to map column names that have a different
name in the import table. Columns that don't match are filled with null or a default
value if it is listed in the **-val_map** option.

Finally one can add a whole new column to an exsisting table that optionally can be
filled with a math expression that references data in other columns of the current row.
Expressions that reference other columns must surround the column with @{...} see
example below.

Examples

The following example appends two rows of static data given as a TCL list of lists
to the given UserTable that has three columns of data.

```
shell> gui_append_utable -name my_table -rows {
 ...      { /top/ModA  0.01  1000 }
 ...      { /top/ModB  0.03  5000 }
 ...   }
```

The following example first creates a TCL list of lists and then appends it to the
given UserTable.

```
shell> set rows {}
shell> ... loop through collection and set vars and append to list...
```

Feedback

```
shell> lappend rows [list $cellname $delay $area]
shell> ...
shell> gui_append_utable -name MyTable -rows $rows
```

The following example appends a given CSV file to an existing table MyTable.
It also uses a column map to map the target column name like (edge) to the column
that matches that data in the CSV file (rise_fall). The columns in the target table ("Extra Col" and sum) would be filled with null but by defining
a value map that maps the target column with a default value it then fills that
column with a default value if it is empty.

```
        set column_map {
  edge  rise_fall
  cap   capacitance
}

set empty_value_map {
  {Extra Col} {not used}
  sum         0
}

gui_append_utable -name MyTable -import values.csv \\
  -val_map $empty_value_map -col_map $column_map
```

Finally below is an example of adding a new expression column to a table.

```
        # Create a table with 3 columns with data type double (postfix)
gui_create_utable -name MyTable -col {Label A:double B:double}

# Add some rows:
gui_append_utable -name MyTable -rows {
  {LabelA 0 1}
  {LabelB 2 3}
  {LabelC 4 5}
}

# Now lets add a new expression column:
# NOTE: referenced column names are surrounded by @{...}
gui_append_utable -name MyTable -column Expr:double -expr { 100 * (@{A} +
 @{B}) }

# Now add a new row (Note: Requires an empty value for the expr column
 added in the previous step!):
```

Feedback

```
gui_append_utable -name MyTable -rows {LabelD 11 22 {}}
```

See Also

- gui_close_utable
- gui_change_selection_utable
- gui_create_utable
- gui_export_utable
- gui_fill_utable
- gui_get_utable
- gui_import_utable
- gui_set_utable_meta
- gui_open_utable
- gui_show_utable
- gui_write_utable

---

gui_change_charts_model

Change data in model used in chart's plots

Syntax

```
status gui_change_charts_model
-model model_id
[ -column string ]
{ -add | -delete | -modify | -calc | -query }
[ -header string ]
[ -type string ]
[ -expr string ]
```

Data Types

model_id int

Arguments

-model *model_id*

Model to change (previously created by *gui_create_charts_model* command).

`-column string`

The column name or number to delete, modify, calculate or query.

`-add`

Add column to model. The *expr* option specifies the expression to use to calculate the values in the new column.

`-delete`

Delete column from model. Use the *column* option to specify the column to delete.

Note: the original columns of the model cannot be deleted, only newly added columns can be removed after they are created.

`-modify`

Modifies the values in an existing column. Use the *column* option to specify the column to modify and the *expr* option to specify the expression to use to calculate the new values in the column.

Note: the original columns of the model cannot be modified, only newly added columns can be modified are created.

`-calc`

Calculate values from model. Use the *column* option to specify the default column for the values and the *expr* option to specify the expression to use to calculate the values.

The calculated values are returned by the command. The model is not modified.

`-query`

Query values from model. Use the *column* option to specify the default column for the values and the *expr* option to specify the expression to use to query the values.

The expression should return true or false and the returned values are those where the expression evaluates to true.

The queried values are returned by the command. The model is not modified.

`-header string`

Specifies the name of the header when adding or modifying a column.

`-type string`

Specify the type of the new column for *add*, *modify* options.

Feedback

`-expr` *string*

Specifies the expression to use when adding, modifying, calculating or querying values of a column of the model.

The expression can use tcl variable names that match the column name (as long as the column name is a legal tcl variable name). A few extra variables are also allowed:

- column : current column
- row : current row
- pi : value of pi (3.141592653...)
- NaN : Special 'not a number' real value that can be used to indicate absence of data.

A number of extra functions are allowed as well as the normal tcl expression functions:

- column : get column value for row
- row : get row value for column
- cell : get cell value for row and column
- setColumn : set column value for row
- setRow : set row value for column
- setCell : set cell value for row and column
- header : get header value for column
- setHeader : set header value for column
- type : get type for column
- setType : set type for column
- map : map row number to range
- bucket : get bucket for column value
- norm : normalize column value
- rand : get random number
- rnorm : get normaized random number
- color : get color from name

Feedback

- remap : map column value to range
- timeval : get time value from column with time format

Some special character sequences in the expression are replaced with model values before the expression is evaluated.

- @<n> : value for specified column number (<n>) for current row.
- @c : column number.
- @r : row number.
- @nc : number of columns.
- @nr : number of rows.
- @v : value for current row and column.
- @{<name>} : value for specified column name (<name>) for current row.
- #{<name>} : column number for specified column name (<name>)

Normally the '@' symbols return data in the value format but this can be forced to be a string by using '@#' instead of '@'.

Description

Change data in existing model to add, delete or modify columns for use in charts.

Examples

The following example adds a new column to the model where the value is the sum of the first two columns.

```
shell> gui_change_charts_model -model $model -add -expr {column(0) +
 column(1)} -header Sum
```

See Also

- gui_create_charts_plot
- gui_define_charts_proc

---

gui_change_error_highlight

Manipulates error objects highlight sets.

Syntax

string *gui_change_error_highlight*

-add | -remove [-color *color_id*] [-selected | -all | -all_visible] [-type *error_type_list*] [-layer *layer_strings*] [-include_net *nets* | -exclude_net *nets*]

```
color_name        A color name
error_type_list   A Tcl list of error type names
layer_strings     A Tcl list of layer strings
nets              A Tcl list of net names or a collection of nets
```

Arguments

`-add | -remove`

Required mutually exclusive option which specifies that the errors are added or removed from the error highlight set. If -add, the specified errors become drawn in the given highlight color. If -remove, the specified errors become drawn in the default violation color.

`-color <color_id>`

This option specifies the color for the operation. If a color is given with the -color option, it must be one of the supported color names: blue, green, light_blue, light_green, light_orange, light_purple, light_red, orange, purple, red, yellow. If -color option is provided with the -add option, the specified color will be used to color the specified errors in the given color. If -color option is provided with the -remove option, errors in the specified color highlight set will be removed from the highlight set and will become drawn in the default violation color. When given with the -remove option, this option is mutually exclusive with -selected, -all, -all_visible, -type, -layer, -include_net, and exclude net options.

`-selected`

This option is mutually exclusive with -all, -all_visible, -type, -layer, -include_net and -exclude_net. If this option is present, currently selected errors in the current tab error list become highlighted in the specified color.

`-all`

This option is only meaningful with the -remove and is mutually exclusive with -color, -all_visible, and -selected. When given with the -remove option, removes highlight colors from all errors in all open error views.

`-all_visible`

This option is mutually exclusive with -selected, -type, -layer, -include_net and -exclude_net. If this option is given with the -add option, currently visible errors in the current tab error list become highlighted in the specified color. If this option is given with the -remove option, all error highlight colors are removed from currently visible errors in the current tab error list. They become drawn in the default violation color.

`-type <error_type_list>`

This option is mutually exclusive with -selected and -all_visible. It may be combined with -layer, and -include_net or -exclude_net. Errors that are of one of the given types become highlighted in the specified color. If combined with layer and/or net specifications, errors must match all given specifications to be selected for highlighting.

`-layer <layer_string_list>`

This option is mutually exclusive with -selected and -all_visible. It may be combined with -type, and -include_net or -exclude_net. Errors that are of one of the given layers become highlighted in the specified color. If combined with error type and/or net specifications, errors must match all given specifications to be selected for highlighting.

`-include_net <nets>`

This option is mutually exclusive with -selected, -all_visible and with -exclude_net. It may be combined with -type and -layer. Errors that are associated with one of the given nets become highlighted in the specified color. NULL net is specified with an empty net name: "". If combined with error type and/or layer specifications, errors must match all given specifications to be selected for highlighting.

`-exclude_net <nets>`

This option is mutually exclusive with -selected, -all_visible and with -include_net. It may be combined with -type and -layer. Errors that are not associated with one of the given nets become highlighted in the specified color. NULL net is specified with an empty net name: "". If combined with error type and/or layer specifications, errors must match all given specifications to be selected for highlighting.

Description

The command specifies the color to use in drawing the specified errors. Errors can be drawn in a highlight color by adding them to the error highlight color set. Errors can be removed from the error highlight color set and revert to being drawn in the default violation color. The command behavior is determined by combinations of the options as follows:

Adding errors to a highlight set:

If -color is given with the -add option, the specified errors are added to the error highlight color set and become draw in the specified color. The specified color becomes the current error highlight color.

If -color is omitted with the -add option, the specified errors are added to the current error highlight color set and become draw in the current error highlight color.

If -selected is given with the -add option, selected errors are added to the error highlight color set and become draw in the specified color.

If -all_visible is given with the -add option, all visible errors in the current tab error list are added to the error highlight color set and become draw in the specified color.

If -type, -layer -include_net or -exclude_net is given with the -add option, visible errors in the current tab error list matching the given criteria are added to the error highlight color set and become drawn in the specified color.

Removing errors from a highlight set:

If -color is given with the -remove option, all errors in the specified error highlight color set are removed and become drawn in the default violation color.

If -selected is given with the -remove option, selected errors are removed from respective error highlight color sets and become drawn in the default violation color.

If -all is given with the -remove option, all errors are removed from respective error highlight color sets and become drawn in the default violation color.

If -all_visible is given with the -remove option, all visible errors in the current tab error list are removed from respective error highlight color sets and become drawn in the default violation color.

If -type, -layer -include_net or -exclude_net is given with the -remove option, visible errors in the current tab error list matching the given criteria are removed from the respective error highlight color set and become drawn in the default violation color.

Examples

The following example puts the currently selected errors in the blue error highlight set:

```
gui_change_error_highlight -add -color blue -selected
```

The following example puts "Short" type errors associated with a net named "C0_9_" in the red error highlight set:

```
gui_change_error_highlight -add -color red -type {Short} -include_net {C0_9_}
```

The following example puts "Floating Port" type errors not associated with NULL net in the green error highlight set. Please note the use of parentheses as delimiters. Since the expected argument for -type and -exclude_net options is a list of strings, multi-word strings or empty string signifying NULL net must be delimited:

```
gui_change_error_highlight -add -color green -type {{Floating Port}} -exclude_net {{}}
```

The following example adds all visible errors in the current tab error list to the light_red highlight set:

```
gui_change_error_highlight -add -color light_red -all_visible
```

Feedback

The following example removes all errors from all error highlight sets:

gui_change_error_highlight -remove -all

The following example removes errors in the red color highlight set from the highlight set:

gui_change_error_highlight -remove -color red

The following example removes all visible errors in the current tab error list from highlight sets:

gui_change_error_highlight -remove -all_visible

See Also

- gui_set_selected_errors

---

gui_change_highlight

Manipulate the set of globally highlighted objects.

Syntax

string *gui_change_highlight*

```
[-add | -remove | -toggle]
[-color color_id | -all_colors]
[-collection clct]

string          color_id
collection      clct
```

Arguments

`-add`

Use -add to highlight a collection of objects in a color specified with color. If -color is not specified, the current color is used. You cannot use -all_colors with -add. You must specify a collection with -collection. If -remove or -toggle are not specified, then -add is implied.

`-remove`

Use -remove to remove highlighting from objects. Use -collection to remove highlighting from specific objects. Use -color to remove highlighting from objects of a specific color. Use -all_colors to remove all highlighting. If you specify a collection, you cannot specify colors.

`-toggle`

Use -toggle to toggle the highlight state of a collection of objects. You must use -collection with -toggle, and you cannot use -color or -all_colors. Toggling an

object that is highlighted will cause it to no longer be highlighted. Toggling an object that is not highlighted with cause it to be highlighted with the current color. If no operation is specified, then -add is assumed.

`-color` *`color_id`*

Specifies the color to be used for the highlight operation. This is mutually exclusive with -all_colors. The allowed colors are blue, light_blue, yellow, purple, light_purple, orange, light_orange, red, light_red, green, and light_green. If neither -color nor -all_colors is specified, then the current highlight color is assumed. You cannot use -color with -toggle.

`-all_colors`

This option is only valid with -remove. Use this option to clear all highlight colors.

Description

The gui_change_highlight command is used to operate on the set of highlighted objects. Objects can be added or removed from the set, or their presence can be toggled as described above. The set has a current color that can be modified with gui_set_highlight_options(2). The current color is used as needed when no color is specified.

The tool provides eleven built in highlight colors with the names yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

The gui_change_highlight command uses colors from the highlight color palette.

Examples

Highlight in green all cells with names matching foo*.

shell> gui_change_highlight -color green -collection [get_cells foo* -hier -all]

Remove highlights from all objects in the collection returned by the get_cells command.

shell> gui_change_highlight -remove -collection [get_cells foo* -hier -all]

Clear all objects with blue highlights.

shell> gui_change_highlight -remove -color blue

Clear objects with the current highlight color.

shell> gui_change_highlight -remove

Clear all highlights.

shell> gui_change_highlight -remove -all_colors

Toggle the highlight state of all objects in a collection returned by the get_cells command.

Feedback

```
shell> gui_change_highlight -toggle -collection [get_cells foo*]
```

See Also

- gui_get_highlight
- gui_get_highlight_options
- gui_set_highlight_options
- gui_get_color_value

---

gui_change_schematic

Performs abstraction on the current or specified schematic view.

Syntax

string *gui_change_schematic*

```
-type abstraction_name
-operation operation_name
-clct objects
[-window window_name]
```

Data Types

```
abstraction_name    string
operation_name      string
objects             collection
window_name         string
```

Arguments

`-type abstraction_name`

Name of abstraction type. Currently supported types: "Hierarchy".

`-operation operation_name`

Name of operation to be applied for the specified abstraction type. Currently supported operations: "Expand", "Collapse".

`-clct objects`

The collection of objects in the target schematic to which the specified abstraction operation will be applied.

`-window window_name`

Optionally specifies the target schematic to which the abstraction is applied. If not specified, the current application schematic will be the target.

Feedback

Description

The command applies the specified abstraction operation to the specified objects within the target schematic view.

Examples

The following example collapses currently selected modules within the current schematic view.

```
prompt> gui_change_schematic -type Hierarchy -operation Collapse -clct
 [get_current_selection]
```

---

gui_change_selection_utable

Change Current Selection by adding objects named in the given table.

Syntax

string *gui_change_selection_utable*

```
-name         Table_Name
[-obj_col     Col_Name]
[-filter      Filter]

Table_Name    String
Col_Name      String
Filter        Table Filter String
```

Arguments

`-name` *Table_Name*

The name of the user table to use for finding object names to add to the current selection.

`-obj_col` *Col_Name*

The column name containing object names with a data type set. By default it uses the full_name and object_class type or named columns.

`-filter` *Filter*

This argument allows you to filter the rows in the given table to only rows that match the filter. The filter expression is the same expression allowed in the GUI filter field of the UserTable GUI view.

Description

```
This command takes a table name and if a filter is given, reduces the
 rows
to those that match the filter and then tries to add to the current
 selection
```

Feedback

any object names found in those table rows. The object names are searched for
in the column given by the *-obj_col* command argument. By default it uses the full_name and object_class type or named columns to determine object names
to add to the current selection.

Examples

The following example adds all object with the object names found in the column 'pins' that match the filter '*ALU*'.

```
shell> gui_change_selection_utable -name my_table -obj_col pins -filter
 {pins =~ *ALU*}
```

See Also

- gui_append_utable
- gui_close_utable
- gui_create_utable
- gui_export_utable
- gui_import_utable
- gui_open_utable
- gui_show_utable
- gui_write_utable

---

gui_clear_error_data_filter

Remove the filter from error data.

Syntax

string *gui_clear_error_data_filter*

Description

Remove the filter and restore visibility of errors. Errors may remain hidden based on null net association or fixed status state if the "Hide Null-Net Errors" option or the "Hide Fixed Errors" option has been set.

Examples

The following example removes the error data filter

gui_clear_error_data_filter

Feedback

See Also

- gui_error_browser
- gui_set_error_data_filter
- gui_set_error_browser_option

---

gui_clear_parasitics

Clears parasitics annotated on a net.

Syntax

status *gui_clear_parasitics*

```
[-all]
[nets]
```

Data Types

```
nets            string
```

Arguments

`-all`

Clears parasitic annotation for all nets. This is the default if no nets are specified.

nets

A list of one or more nets for which to clear parasitics. Glob-style wildcards (*) are supported.

The default is to clear all nets.

---

gui_clear_selected_errors

Clears selection in the Error Browser.

Syntax

string *gui_clear_selected_errors*

Description

Clears selected errors in the Error Browser.

Feedback

Examples

The following example clears selected errors:

gui_clear_selected_errors

See Also

- gui_error_browser
- gui_get_errors
- gui_set_current_errors
- gui_set_selected_errors

---

gui_close_error_data

Removes an error data file from the error browser.

Syntax

```
status gui_close_error_data
[error_data]
```

Data Types

```
error_data        collection
```

Arguments

error_data

A collection of physical DRC error data objects to remove from the error browser. A collection of the error data or the error data name can be used to remove it from the error browser.

Description

The *gui_close_error_data* command removes the given error data from the GUI error browser. If no error data is given, then the command removes all error data from the error browser and hides the error browser.

When a command opens an error data, it increments the open_count for an error data. Incrementing the open_count needs to be balanced by the same number of decrementing the open_count before the error data is closed and removed from memory.

The *gui_open_error_data* command opens an error data if necessary. It does not increment the open_count if the error data is already open and in memory. If the *gui_open_error_data* command increments the open_count, then a subsequent call to *gui_close_error_data* for the same error data decrements the open_count. Otherwise,

Feedback

a subsequent call to *gui_close_error_data* simply removes the error_data from the error browser without decrementing the open_count.

Examples

The following example removes the physical DRC error data file that is named "my_design_dppinassgn.err" from the error browser.

```
prompt> gui_close_error_data "my_design_dppinassgn.err"
1
```

The following example removes all physical DRC error data file from the error browser and hides the error browser.

```
prompt> gui_close_error_data
1
```

See Also

- gui_open_error_data
- gui_get_error_data
- close_drc_error_data
- open_drc_error_data
- save_drc_error_data
- get_drc_error_data
- create_drc_error_data
- remove_drc_error_data
- collections

---

gui_close_utable

Close and free the given list of UserTables in memory.

Syntax

string *gui_close_utable*

```
-names   Table_Names

Table_Names     String List
```

Feedback

Arguments

`-names Table_Names`

The list of table names to close and free in memory.

Description

```
This command closes and frees all the table names given by the user. If
 the
table was in read only mode (streaming), then the connection to the table
file is closed.
```

Examples

The following example closes a list of user tables.

```
shell> gui_close_utable -names {my_table1 my_table2}
```

See Also

- gui_append_utable
- gui_change_selection_utable
- gui_create_utable
- gui_export_utable
- gui_import_utable
- gui_open_utable
- gui_show_utable
- gui_write_utable

---

gui_close_window

Close the specified window

Syntax

```
status gui_close_window
{ -window window_id | -type window_type | -all }
[ -exact_type_match ]
```

Data Types

```
window_id       string
window_type     string
```

Feedback

Arguments

`-window window_id`

Specifies the window name id.

Any matching window of this name will be closed.

This option precludes the *-type* and *-all* options.

`-type window_type`

Specifies the window type id. Any matching window of this type will be closed.

If the *-exact_type_match* options is not specified then types derived from this window type will also be closed. Derived types are created with the *gui_create_window_type* command.

This option precludes the *-window* and *-all* options.

`-all`

Specifies that we want to close all the windows.

This option precludes the *-window* and *-type* options.

`-exact_type_match`

Specifies that types derived from the specified type are excluded when the *-type* option is specified.

Note: this option should only be used with the *-type* option.

Description

This command closes the specified window(s).

Windows can be specified by name, type or as all windows.

Examples

The following examples will create a toplevel window and a child layout view window, then close the command layout window.

```
shell> set top [gui_create_window -type TopLevel]
shell> set x [gui_create_window -type Layout -parent $top]
shell> gui_close_window -window $x
```

See Also

- gui_create_window
- gui_exist_window

Feedback

- gui_show_window
- gui_get_window_types
- gui_get_window_ids
- gui_set_active_window
- gui_get_current_window

---

gui_connect_charts_signal

Connect to charts signal

Syntax

```
status gui_connect_charts_signal
{ -view view_name | -plot plot_name }
-from signal_name
-to tcl_proc
```

Data Types

```
view_name    string
plot_name    string
signal_name  string
tcl_proc     string
```

Arguments

`-view view_name`

Charts view to connect to.

`-plot plot_name`

Plot to connect to.

`-from signal_name`

Signal name to connect to.

Supported names are:

objIdPressed (id proc) plot object clicked on with mouse

annotationIdPressed (id proc) plot or view annotation clicked on with mouse

chartObjsAdded (non-id proc) objects added to chart

`-to tcl_proc`

Tcl procedure to call when the signal is emitted.

Feedback

For plot an id procedure is called with three arguments (view, plot and object_id) and a non-id procedure is called with two arguments (view, chart).

For view an id procedure is called with two arguments (view, object_id) and a non-id procedure is called with one arguments (view).

Description

Connects signals emitted from charts events to tcl callbacks.

Examples

The following example calls chartObjPressed proc when object in plot pressed.

```
shell> gui_connect_charts_signal -plot $plot -from objIdPressed -to
 chartObjPressed
```

See Also

- gui_create_charts_plot

---

gui_create_attrgroup

Creates a group of attributes for an object type.

Syntax

```
status gui_create_attrgroup
-class design_object
-name name
[-attr_list {{attr_1}...{attr_n}}]
```

Data Types

```
design_object      string
name               string
attr_1             string
attr_n             string
```

Arguments

`-class design_object`

Specifies the type of design object.

`-name name`

Specifies the name of the attribute group to be created for the specified object type.

`-attr_list {{attr_1}...{attr_n}}`

Lists and orders the attributes to be included in the group.

Feedback

Description

The *gui_create_attrgroup* command creates a new attribute group for a specific type of design object. The command adds the attribute group to the display in GUI tools, such as the Properties dialog box, List views, layout view InfoTips, and other graphic windows.

An attribute can be assigned to more than one attribute group at the same time.

The order of the attributes in an attribute group is important and is set by the *-attr_list* option when you create the group. Use the *gui_update_attrgroup* command if you need to change or reorder the attribute list for an attribute group.

Examples

```
prompt> gui_create_attrgroup -class Cell -name "Hierarchy" \\
   -attr_list { "Name" "Hierarchical" "Owning Cell"
   "isPhysConstraint" "Hard Macro" "Num Children" "Num Nets"
   "Num Pins" "Utilization" "Area" "Aspect Ratio" "Min Aspect"
   "Max Aspect" "Min Area" "Max Area" "Area / 1M" "Hier Req Area / 1M"
   "Reference" "Logical Name" "FullName" }

prompt> gui_create_attrgroup -class Cell -name "Edit" \\
   -attr_list { "Name" "isPhysConstraint" "Orientation"
   "Location" "Bounding Box" "Fixed Location" "Is Resizable"
   "Hard Macro" "Placed" "Is IO" "Off Litho Grid" "FullName" }

prompt> gui_create_attrgroup -class Pin -name "Timing" \\
   -attr_list { "Name" "Is Clock" "Slew"  "slack_max" }

prompt> gui_create_attrgroup -class Net -name "Timing" \\
   -attr_list { "Name" "Lumped C" "Total Lumped C"
   "Lumped Resistance" }
```

See Also

- gui_delete_attrgroup
- gui_list_attrgroups
- gui_update_attrgroup

---

gui_create_category_rule

Creates a category rule for automatic generation of one or more object categories.

Syntax

string *gui_create_category_rule*

```
[-name rule_name]
[-filter filter_spec]
```

Feedback

```
-category category_spec
[-builtin]
```

Data Types

```
rule_name           string
filter_spec         string
category_spec       string
```

Arguments

`-name rule_name`

Specifies the name of the category rule to be created. If this option is omitted, a unique new rule name is automatically generated.

`-filter filter_spec`

Specifies the filter. The basic form of a filter conditional expression is a series of relations joined together with && (and), || (or), and ! (not) operators. Parentheses are used to override precedence. The basic relation is either a Boolean attribute or a comparison of one non-Boolean attribute with another or with a constant value using relational operators.

For example, a filter expression can have any of the following forms:

- *-filter {endpoint_clock_is_propagated}* (Boolean attributes)
- *-filter {(slack < 0)}* (Comparison with a literal number)
- *-filter {(path_group.full_name == "inputs")}* (Comparison with a literal string)
- *-filter {(startpoint_clock.full_name == endpoint_clock.full_name)}* (Comparison of attributes)

Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection. The following relational operators are supported:

- == (Equal)
- *!*= (Not equal)
- =~ (Matches pattern)
- *!*~ (Does not match pattern)
- &~ (Contains all elements)
- |~ (Contains some elements)
- > (Greater than)

- < (Less than)
- >= (Greater than or equal to)
- <= (Less than or equal to)

The matching operators are used to match wildcard regular expressions. The containment operators are used to compare space-separated set or list attributes.

`-category` *`category_spec`*

Specifies the category. The category specification can be fixed literal text, or it can include an optional object attribute name enclosed in angle brackets; for example, *-category <path_type>*. Additional literal text is optionally allowed to the left and to the right of the angle brackets; for example *-category {Path Type: <path_type>}*. Some object attributes have values that are object collections of size one. For those attributes, "dot notation" can be used to access attributes of the object in the collection; for example, *-category <startpoint_clock.full_name>*.

When a category rule is "executed" later in the GUI, a separate category (bin of objects) is generated for each unique value of the attribute specified by the *-category* option. (The GUI "category-rule-execution engine" examines attribute values for an input set of objects.)

`-builtin`

Indicates that the rule being created belongs to the built-in group of category rules. If this option is omitted, the rule being created does not belong to the built-in group.

The built-in group of category rules is a group of category rules that are created before any non-built-in rules. This group of rules includes both built-in rules created by the tool (at tool startup time) and built-in rules that you create before creating any non-built-in rules.

Description

This command creates a category rule. If rule creation is successful, the command returns the name of the newly-created category rule; otherwise, it returns an empty string. After a category rule has been created, it can be used later (at "category-rule execution time" in the GUI) with other rules to generate one or more object categories.

Examples

This example creates a simple category rule:

```
prompt> gui_create_category_rule -name pathgroups \\
        -category <path_group.full_name>
pathgroups
```

Feedback

This example creates a simple filtering rule:

```
prompt> gui_create_category_rule -name {Negative Slack} \\
        -category {Failing Paths} -filter {slack < 0}
Negative Slack
```

This example creates a regular expression filtering rule:

```
prompt> gui_create_category_rule -name {ATPG Sessions} \\
        -category {ATPG Sessions} -filter {session_name =~ "atpg*"}
Negative Slack
```

This example creates a set or list containment filtering rule:

```
prompt> gui_create_category_rule -name {Through CPU and CTRL} \\
        -category {Through CPU and CTRL} -filter { blocks &~ "CPU CTRL"}
Negative Slack
```

See Also

- gui_list_category_rules
- gui_remove_category_rules

---

gui_create_charts_arrow_annotation

Create arrow annotation in view or plot

Syntax

```
arrow_id gui_create_charts_arrow_annotation
{ -view view_name | -plot plot_name }
[ -id string ]
[ -tip string ]
-start position
-end position
[ -line_width length ]
[ -fhead string ]
[ -thead string ]
[ -angle angles ]
[ -length lengths ]
[ -properties name_value_list ]
```

Data Types

```
view_name        string
plot_name        string
position         string
length           string
angles           string
lengths          string
```

Feedback

```
name_value_list  string
arrow_id         annotation_id
```

Arguments

`-view view_name`

Charts View to add annotation to.

`-plot plot_name`

Plot to add annotation to.

`-id string`

Optional identifier string for annotation.

`-tip string`

Optional tip string for annotation.

`-start position`

Start point of the arrow line.

`-end position`

End point of the arrow line.

`-line_width length`

Width of the connecting line.

`-fhead string`

Front arrow head type.

One of triangle, stealth, diamond, line or none.

`-thead string`

Tail arrow head type.

One of triangle, stealth, diamond, line or none.

`-angle angles`

Angle of arrow heads.

Use single value for same angle for both front and tail heads. Use two values for different angle for front and tail heads.

`-length lengths`

Length of arrow heads.

Use single value for same length for both front and tail heads. Use two values for different length for front and tail heads.

Feedback

`-properties name_value_list`

Specifies a list of name value pairs for the properties to set on the created annotation.

Returns

arrow_id

Returns id of created annotation.

Description

Add an arrow annotation to plot or view.

The arrow annotation is a line with optional arrow heads draw at each end.

The line and arrow heads can be customized.

Examples

The following example creates an arrow annotation on a plot from (0 0) to (100 100) with arrows at both ends. The front head uses a line and the tail head a triangle.

```
shell> gui_create_charts_arrow_annotation -plot $plot -start {0 0} -end
 {100 100} \
  -fhead line -thead triangle
```

See Also

- gui_create_charts_plot
- gui_create_charts_ellipse_annotation
- gui_create_charts_point_annotation
- gui_create_charts_polygon_annotation
- gui_create_charts_polyline_annotation
- gui_create_charts_rectangle_annotation
- gui_create_charts_text_annotation
- gui_remove_charts_annotation
- gui_set_charts_property
- gui_get_charts_property

gui_create_charts_ellipse_annotation

Create ellipse annotation in view or plot

Feedback

Syntax

```
ellipse_id gui_create_charts_ellipse_annotation
{ -view view_name | -plot plot_name }
[ -id string ]
[ -tip string ]
-center position
-rx length
-ry length
[ -properties name_value_list ]
```

Data Types

```
view_name         string
plot_name         string
position          string
length            float
name_value_list   string
ellipse_id          annotation_id
```

Arguments

`-view view_name`

Charts View to add annotation to.

`-plot plot_name`

Plot to add annotation to.

`-id string`

Optional identifier string for annotation.

`-tip string`

Optional tip string for annotation.

`-center position`

Center of the ellipse.

`-rx length`

X Radius of the ellipse.

`-ry length`

Y Radius of the ellipse.

`-properties name_value_list`

Specifies a list of name value pairs for the properties to set on the created annotation.

Feedback

Returns

ellipse_id

Returns id of created annotation.

Description

Add an ellipse annotation to plot or view.

The background and border of the ellipse can be customized.

Examples

The following example creates an ellipse annotation on a plot centered at (0 0) with a radius of 60 in x direction and 40 in y direction.

```
shell> gui_create_charts_ellipse_annotation -plot $plot -center {0 0} -rx
 60 -ry 40
```

See Also

- gui_create_charts_plot
- gui_create_charts_arrow_annotation
- gui_create_charts_point_annotation
- gui_create_charts_polygon_annotation
- gui_create_charts_polyline_annotation
- gui_create_charts_rectangle_annotation
- gui_create_charts_text_annotation
- gui_remove_charts_annotation
- gui_set_charts_property
- gui_get_charts_property

---

gui_create_charts_model

Create a model from data for use in charts

Syntax

```
model_id gui_create_charts_model
{ -csv filename | -clct collection |
-tcl tcl_list }
[ { -comment_header | -first_line_header } ]
[ -first_column_header ]
```

Feedback

```
[ -transpose ]
[ -columns column_names ]
[ -column_type column_type ]
[ -name string ]
```

Data Types

```
filename      string
collection    string
tcl_list      string
column_names  string
column_type   string
```

Arguments

`-csv filename`

Create model from contents of specified csv filename.

A CSV file has comma separated fields.

The *comment_header*, *first_line_header* and *first_column_header* options can be used to determine which (if any) lines are used for horizontal and vertical headers.

`-clct collection`

Create model from contents of a collection. Each row consists of a list of attribute values from the collection object.

The *columns* option can be used to specified which attributes are used to populate the table from the collection. If the *columns* option is not specified then the attributes from the ALL attribute group are used.

`-tcl tcl_list`

Create model from the values in a tcl variable.

The tcl variable must be a list of lists. The data is a list of rows where each row is a list of column values.

The *transpose* option can be used to transpose the value array so lists of columns can be specified where each column is a list of row values.

`-comment_header`

Specifies that the first comment (line starting with a #) contains the names of the horizontal column headers.

Only used by the *csv* option.

`-first_line_header`

Specifies that the first line contains the names of the horizontal column headers.

Feedback

Used by the *csv* and *tcl* options.

`-first_column_header`

Specifies that the first column (first field in each row) contains the names of the vertical row headers.

If the data also has a horizontal header then the first field of the horizontal header is ignored.

Used by the *csv* and *tcl* options.

`-transpose`

Specifies that the tcl value array from *tcl* option is transposed so lists of columns can be specified where each column is a list of row values.

`-columns` *`column_names`*

Specifies a list of names for the attributes to query from the collection to generate the columns of each row.

Used by the *clct* option.

`-column_type` *`column_type`*

Specifies the type and optional formatting of the values in the columns.

If the type of the column is not specified then the application will try to automatically determine it depending on whether the values can be all converted to real or integer values. If not the default string type is used.

The specified value is an ordered list of column type data. Each column type data is a list where the first element specifies the type and any extra elements specify extra name values to customize the type. By default, the type is associated with the column number matching the index of the item but a specific column can be specified by including the column name or number with the type value in the first element.

Syntax:

- column_type_data : { <column_type> [<name_values>]}
- column_type : { [<column>] <type> }
- column : column_name|column_index
- type : integer|real|string|...
- name_values : <name_value> <name_value> ...
- name_value : { <name> <value> }

Feedback

The supported name values are dependant on the column type.

Column types can be queried using:

```
gui_get_charts_data -global -name column_types
```

The column types named values can be queried using:

```
gui_get_charts_data -global -name column_type.names -data $type
```

`-name` *`string`*

Specifies the name of the model.

This can be used to annotate the model so the user can more easily identify what data it contains in the gui.

If the *csv* option is used then the filename is the default name.

Returns

model_id

The id of the created model.

This can be used as input to the *gui_create_charts_plot* command and other commands to do further processing on the model.

Description

This command creates a persistant model and loads data into it so it can be used as the data source for one or more plots.

The command returns a model id which can be used as input to the *gui_create_charts_plot* and other charts commands.

Examples

The following example creates a model from the csv file "test.csv" using the first line for the column headers.

```
shell> gui_create_charts_model -csv "test.csv" -first_line_header
```

The following example creates a model from a collection of all shapes with columns for the name and layer attributes.

```
shell> gui_create_charts_model -clct [get_nets] -columns [list name
 layer]
```

See Also

- gui_create_charts_plot

Feedback

gui_create_charts_plot

Create a plot of the type using the supplied data

Syntax

```
plot_name gui_create_charts_plot
-view view_name
{ -clct collection | -model model_id }
-type plot_type
[ -columns column_names ]
[ -title string ]
[ -properties name_value_list ]
[ -xmin float ]
[ -ymin float ]
[ -xmax float ]
[ -ymax float ]
```

Data Types

```
view_name        string
collection       string
model_id         int
plot_type        string
column_names     string
name_value_list  string
```

Arguments

`-view view_name`

Parent view for plot.

`-clct collection`

Collection to use for input data. Each row consists of a list of attribute values from the collection object.

The *columns* option is used to specified which attributes are read from the collection to populate the table values.

Note: the resultant model is a snapshot of the specified collection's attribute values and will not updates as the collection is changed.

`-model model_id`

The model to use for input data.

`-type plot_type`

The type of plot to display:

The plot types can be queried using the command *gui_get_charts_data -global -name plot_types*

Feedback

`-columns column_names`

Specifies the columns to use for each plot column parameter.

The columns string is a command separated list of name value pairs of the form "<parameter name>=<column name>".

The parameter names are plot type dependent.

When the *model* option is specified the column names are column indices (0 based) or column header names.

When the *clct* option is specified the column names are attributes names.

`-title string`

Specifies the title for the plot.

`-properties name_value_list`

Specifies a list of name value pairs for the properties to set on the created plot.

`-xmin float`

User specified minimum x value (to override the auto calculated range).

`-ymin float`

User specified minimum y value (to override the auto calculated range).

`-xmax float`

User specified maximum x value (to override the auto calculated range).

`-ymax float`

User specified maximum y value (to override the auto calculated range).

Returns

plot_name

The id of the created plot.

Description

Create plot in a view using values in data model.

The view and model data must be created before the plot can be added.

Examples

The following example creates a distribution plot from the specified model using values from the first column.

Feedback

```
shell> set window [gui_get_current_window -mru]
shell> set view [gui_create_window -parent $window -type Charts]
shell> gui_create_charts_plot -view $view -model $model -type
 distribution -columns "value=0"
```

See Also

- gui_create_window
- gui_create_charts_model

---

gui_create_charts_point_annotation

Create point annotation in view or plot

Syntax

```
point_id gui_create_charts_point_annotation
{ -view view_name | -plot plot_name }
[ -id string ]
[ -tip string ]
-position position
[ -size length ]
[ -type symbol_type ]
[ -properties name_value_list ]
```

Data Types

```
view_name        string
plot_name        string
position         string
length           float
symbol_type      string
name_value_list  string
point_id           annotation_id
```

Arguments

`-view view_name`

Charts View to add annotation to.

`-plot plot_name`

Plot to add annotation to.

`-id string`

Optional identifier string for annotation.

`-tip string`

Optional tip string for annotation.

`-position` *position*

Point position.

`-size` *length*

Point size.

`-type` *symbol_type*

Point symbol type.

One of: dot, cross, plus, y, triangle, itriangle, box, diamond, star5, star6, circle, pentagon, ipentagon, hline, vline.

`-properties` *name_value_list*

Specifies a list of name value pairs for the properties to set on the created annotation.

Returns

point_id

Returns id of created point.

Description

Add a point annotation to plot or view.

The background and border of the point can be customized.

Examples

The following example creates a point annotation on a plot with a circle symbol at (0 0).

```
shell> gui_create_charts_point_annotation -plot $plot -position {0 0}
 -type circle
```

See Also

- gui_create_charts_plot
- gui_create_charts_arrow_annotation
- gui_create_charts_ellipse_annotation
- gui_create_charts_polygon_annotation
- gui_create_charts_polyline_annotation
- gui_create_charts_rectangle_annotation
- gui_create_charts_text_annotation

Feedback

- gui_remove_charts_annotation
- gui_set_charts_property
- gui_get_charts_property

---

gui_create_charts_polygon_annotation

Create polygon annotation in view or plot

Syntax

```
poly_id gui_create_charts_polygon_annotation
{ -view view_name | -plot plot_name }
[ -id string ]
[ -tip string ]
-points position_list
[ -properties name_value_list ]
```

Data Types

```
view_name         string
plot_name         string
position_list     string
name_value_list   string
poly_id             annotation_id
```

Arguments

`-view view_name`

Charts View to add annotation to.

`-plot plot_name`

Plot to add annotation to.

`-id string`

Optional identifier string for annotation.

`-tip string`

Optional tip string for annotation.

`-points position_list`

List of polygon points.

`-properties name_value_list`

Specifies a list of name value pairs for the properties to set on the created annotation.

Feedback

Returns

poly_id

Returns id of created polygon.

Description

Add a polygon annotation to plot or view.

The background and border of the polygon can be customized.

Examples

The following example creates a polygon annotation on a plot with a red background with 50% alpha.

```
shell> gui_create_charts_polygon_annotation -plot $plot \
  -points "{0 0} {10 10} {20 40} {30 20} {40 10} {50 60} {60 40}" \
  -background 1 -fill_color red -fill_alpha 0.5
```

See Also

- gui_create_charts_plot
- gui_create_charts_arrow_annotation
- gui_create_charts_ellipse_annotation
- gui_create_charts_point_annotation
- gui_create_charts_polyline_annotation
- gui_create_charts_rectangle_annotation
- gui_create_charts_text_annotation
- gui_remove_charts_annotation
- gui_set_charts_property
- gui_get_charts_property

---

gui_create_charts_polyline_annotation

Create polyline annotation in view or plot

Syntax

```
poly_id gui_create_charts_polyline_annotation
{ -view view_name | -plot plot_name }
[ -id string ]
[ -tip string ]
```

Feedback

```
-points position_list
[ -properties name_value_list ]
```

Data Types

```
view_name         string
plot_name         string
position_list     string
name_value_list   string
poly_id             annotation_id
```

Arguments

`-view view_name`

Charts View to add annotation to.

`-plot plot_name`

Plot to add annotation to.

`-id string`

Optional identifier string for annotation.

`-tip string`

Optional tip string for annotation.

`-points position_list`

List of polygon points.

`-properties name_value_list`

Specifies a list of name value pairs for the properties to set on the created annotation.

Returns

poly_id

Returns id of created poly line.

Description

Add a multi-point line annotation to plot or view.

The background and border of the line can be customized.

Examples

The following example creates a polyline annotation on a plot with a red border with 50% alpha.

Feedback

```
shell> gui_create_charts_polygon_annotation -plot $plot \
  -points "{0 0} {10 10} {20 40} {30 20} {40 10} {50 60} {60 40}" \
  -border 1 -stroke_color red -stroke_alpha 0.5
```

See Also

- gui_create_charts_plot
- gui_create_charts_arrow_annotation
- gui_create_charts_ellipse_annotation
- gui_create_charts_point_annotation
- gui_create_charts_polygon_annotation
- gui_create_charts_rectangle_annotation
- gui_create_charts_text_annotation
- gui_remove_charts_annotation
- gui_set_charts_property
- gui_get_charts_property

---

gui_create_charts_rectangle_annotation

Create rectangle annotation in view or plot

Syntax

```
rect_id gui_create_charts_rectangle_annotation
{ -view view_name | -plot plot_name }
[ -id string ]
[ -tip string ]
[ -rectangle string ]
[ -properties name_value_list ]
```

Data Types

```
view_name        string
plot_name        string
name_value_list  string
rect_id            annotation_id
```

Arguments

`-view view_name`

Charts View to add annotation to.

Feedback

`-plot plot_name`

Plot to add annotation to.

`-id string`

Optional identifier string for annotation.

`-tip string`

Optional tip string for annotation.

`-rectangle string`

Rectangle bounding box

`-properties name_value_list`

Specifies a list of name value pairs for the properties to set on the created annotation.

Returns

rect_id

Returns id of created rectangle.

Description

Add a rectangle annotation to plot or view.

The background and border of the rectangle can be customized.

Examples

The following example creates a rectangle annotation on a plot from (0 0) to (100 100).

```
shell> gui_create_charts_rectangle_annotation -plot $plot -start {0 0}
 -end {100 100}
```

See Also

- gui_create_charts_plot
- gui_create_charts_arrow_annotation
- gui_create_charts_ellipse_annotation
- gui_create_charts_point_annotation
- gui_create_charts_polygon_annotation
- gui_create_charts_polyline_annotation
- gui_create_charts_text_annotation

Feedback

- gui_remove_charts_annotation
- gui_set_charts_property
- gui_get_charts_property

---

gui_create_charts_text_annotation

Create text annotation in view or plot

Syntax

```
text_id gui_create_charts_text_annotation
{ -view view_name | -plot plot_name }
[ -id string ]
[ -tip string ]
{ -position position | -rectangle bounding_box }
-text string
[ -properties name_value_list ]
```

Data Types

```
view_name        string
plot_name        string
position         string
bounding_box     string
name_value_list  string
text_id            annotation_id
```

Arguments

`-view view_name`

Charts View to add annotation to.

`-plot plot_name`

Plot to add annotation to.

`-id string`

Optional identifier string for annotation.

`-tip string`

Optional tip string for annotation.

`-position position`

The position of the text.

`-rectangle bounding_box`

The bounding box of the text.

Feedback

`-text` *`string`*

The text to display.

`-properties` *`name_value_list`*

Specifies a list of name value pairs for the properties to set on the created annotation.

Returns

text_id

Returns id of created text.

Description

Add a text annotation to plot or view.

The background, border and font of the text can be customized.

Examples

The following example creates a text annotation on a plot at (0 0) with the text "Test text".

```
shell> gui_create_charts_text_annotation -plot $plot -position {0 0}
 -text "Test text"
```

See Also

- gui_create_charts_plot
- gui_create_charts_arrow_annotation
- gui_create_charts_ellipse_annotation
- gui_create_charts_point_annotation
- gui_create_charts_polygon_annotation
- gui_create_charts_polyline_annotation
- gui_create_charts_rectangle_annotation
- gui_remove_charts_annotation
- gui_set_charts_property
- gui_get_charts_property

---

gui_create_clock_graph

Creates a clock graph.

Feedback

Syntax

gui_create_clock_graph

```
-clct objects
[-levelized]
[-latency]
```

Arguments

`-clct`

Collection of objects to be used to generate a clock graph.

`-levelized`

Generates a levelized clock graph.

`-latency`

Generates a latency clock graph.

Description

The command generates a levelized or latency clock graph using a selected set of objects.

Examples

The following example generates a levelized clock graph using a collection of clock paths.

```
prompt> gui_create_clock_graph -clct $clock_paths -levelized
```

See Also

- change_selection

---

gui_create_menu

Create a menu item for the toplevel menubar or a context menu item. A menu hierarchy, based on the menu name, may be created as necessary to host the menu item.

Syntax

string *gui_create_menu* -menu *HierMenuName*

```
<-tcl_cmd       TclCmd|-separator|-heading headingText>
[-enable_cmd    EnableCmd]
[-get_state_cmd GetStateCmd]
[-icon          IconFilePath]
[-hot_key       HotKey]
[-tooltip       ToolTip]
[-help_string   HelpStr]
[-anchor_item   AnchorItem]
```

```
[-anchor_offset AnchorOffset]
[-position      MenuPos]
[-window_type   WindowTypeName]
[-root          ContextMenuRoot]
[-reference     RefMenuItem]

HierMenuName    String
TclCmd          String
EnableCmd       String
GetStateCmd     String
IconFilePath    String
HotKey          String
ToolTip         String
HelpStr         String
AnchorItem      String
AnchorOffset    Integer
MenuPos         Integer
WindowTypeName  String
ContextMenuRoot String
RefMenuItem     String
```

Arguments

`-menu` *HierMenuName*

HierMenuName specifies the hierarchical menu name of the menu item to create. If a level of the hierarchy does not exist in the menu structure it will be created automatically by this command. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do so.

`-tcl_cmd` *TclCmd*

TclCmd specifies the tcl command to execute when the menu item is selected. This option is mutually exclusive with the -separator and -heading options.

`-separator`

This option creates a menu item separator. It is mutually exclusive with the -tcl_cmd and -heading options.

`-heading` *headingText*

This option creates a text heading in the menu with the specified *headingText*. It is visually distinct from actual menu items, and it is not selectable like an actual menu item is. This is used to provide general grouping information without requiring an additional level of sub menu. This option is mutually exclusive with the -separator and -tcl_cmd options.

`-enable_cmd` *`EnableCmd`*

EnableCmd specifies the tcl command to evaluate whether the menu item should be enabled or disabled. The tcl command should return "true" or "1" to enable, and return "false" or "0" to disable. If the enable_cmd is not specified then the menu item will always be enabled.

`-get_state_cmd` *`GetStateCmd`*

GetStateCmd specifies the tcl command that will be executed to determine whether the menu item should be in an on (checked or pushed-in) or off (unchecked or normal) state. Most menu items don't have persistent state, and for those the get_state_cmd should not be specified. This tcl command returns "true" or "1" for the "on" state, and returns "false" or "0" for the "off" state.

`-icon` *`conFilePath`*

IconFilePath specifies the absolute pathname of the icon file. The file should be in *.xpm* format. Typical application-supplied icons are 16x16 pixels in size and use a transparent background (color 'None' in xpm format).

`-hot_key` *`HotKey`*

HotKey specifies the hotkey (accelerator) for the menu item.

`-tooltip` *`ToolTip`*

ToolTip specifies the tooltip for the menu item.

`-help_string` *`HelpStr`*

HelpStr specifies the help string for the menu item. The help string is displayed in the status bar.

`-anchor_item` *`AnchorItem`*

AnchorItem specifies the existing menu item that will be used as the anchor position for this new menu item. This option is used together with the -anchor_offset option to determine the location of the new menu item. The -position option cannot be used along with the -anchor_item and -anchor_offset options.

`-anchor_offset` *`AnchorOffset`*

AnchorOffset specifies the offset from the anchor item position that will be used to determine the location of the new menu item in the menu hierarchy. The offset must be a positive or negative integer, and not zero. If positive, the new menu item will be placed below the anchor item by the specified offset. If negative, it will be placed above the anchor item position by the absolute value of the specified offset. The -position option cannot be used along with the -anchor_item and -anchor_offset options.

`-position` *`MenuPos`*

MenuPos specifies the absolute menu position for the menu item within the parent popup menu. The -position option cannot be used with the -anchor_item and -anchor_offset options.

`-window_type` *`WindowTypeName`*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

`-root` *`ContextMenuRoot`*

ContextMenuRoot specifies the context menu root to which the menu item being defined will be added. A context menu root groups menu items that share the same context menu root. The context menu root is used for context menu. Context menu, is usually brought up by right clicking on a window and is used to provide context-sensitive features. (Note the difference between a context menu root and the menu root use by a toplevel window's menu bar .)

`-reference` *`RefMenuItem`*

RefMenuItem specifies the menu item under the toplevel window's menu bar, which will be used to define the behavior of the menu item being defined. Often used with the -root option to help associate a context menu item with an already defined menu entry under the menu bar. Used with the -window_type option to completely specify the reference menu item under the menu bar. If the -window_type option is not specified, the default toplevel window type as specified with the tcl variable "::gui_default_window_type" will be used.

Description

This command creates a menu item for the menu bar of a top level window. A menu hierarchy, based on the menu name, may be created as necessary to host the menu item. The menu hierarchy separator is denoted by "->".

These settings are stored either in a user-specific setup file or one that is shared across the installation. The gui initialization sequence will load builtin system configuration, and then apply the installation specific customizations, and finally the user's customizations.

The installation specific setup file is specified by the variable gui_custom_setup_file and will have a default empty value. This can be overridden in the .synopsys_dc.setup file if it is desired. If the file specified by this variable exists then it will be loaded after the gui is initialized.

The user setup file is ~/.synopsys_icc_gui.tcl for IC Compiler.

Feedback

Examples

The following simple example adds a new toplevel popup menu *&Favorite* in the menubar and add a submenu named *Preferences* then add an item named *My &Item* to that submenu. This menu item will be enabled if *enable_my_item* is set to 1 and disabled otherwise. A hotkey and tooltip are also provided.

```
set enable_my_item 1
proc enable_my_item {} {
    if { $::enable_my_item == 1 } {
        return 1
    }
    return 0
}
gui_create_menu -menu "&Favorite->Preferences->My &Item" \\
             -tcl_cmd "echo {executing My Item}" \\
             -enable_cmd "enable_my_item" \\
             -hot_key "Ctrl+N" \\
             -tooltip "short tooltip description for My Item" \\
             -help_string "long status bar description for My Item" \\
             -icon_file "/syn/sparc/my_item.xpm"
```

The following IC Compiler example illustrates creating a new menu item in the context menu of the layout window which will refer to the Zoom Fit Selection menu item from the Layout window main menu bar. This assumes that the tcl variable "gui_default_window_type" is set to the ICC window type "LayoutWindow" and thus it is not necessary to specify the -window_type option with the value of "LayoutWindow".

gui_create_menu -menu "Zoom Fit Selection" \\ -root layout_view_menu_root \\ -reference "View->Zoom->Zoom Fit Selection"

The following example create a new Tcl-based menu entry in the context menu which invokes a procedure # called do_my_thing.

gui_create_menu -menu "Do My Thing" \\ -root layout_view_menu_root \\ -tcl "do_my_thing"

The following IC Compiler example illustrates adding a context menu item to the path schematic context menu which refers to a menu item in the Main window. This example assumes that there is a context menu root called "path_schematic_contextmenu_root", a hierarchical menu item in the toplevel menu bar of the window type "MainWindow" with the hierarchical menu name "Schematic->Delete Selected".

gui_create_menu -menu "Remove Selected" \\ -root path_schematic_contextmenu_root \\ -reference "Schematic->Delete Selected" -window_type MainWindow

Feedback

See Also

- gui_set_hotkey
- gui_report_hotkeys
- gui_delete_menu
- gui_create_toolbar
- gui_delete_toolbar
- gui_create_toolbar_item
- gui_delete_toolbar_item
- gui_get_menu_roots
- gui_default_window_type
- gui_custom_setup_files

---

gui_create_pref_category

Create a preference category.

Syntax

string *gui_create_pref_category* -category *Category*

```
Category        String
```

Arguments

```
-category Category
```

Category specifies the category to create.

Description

This command creates a preference service category with the specified name, which can be used to store and categorize key-value pairs that are created with the *gui_create_pref_key* command. If successful, the command returns the name of the category that is created.

Examples

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false.

Feedback

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type bool
 -value false
```

See Also

- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_create_pref_key

Create a preference key.

Syntax

string *gui_create_pref_key* -key *Key*

```
-value_type ValueType
-value Value
[-category Category]
[-keep_value_if_exist]
[-read_only]
[-description Description]
[-min Minimum_Value]
[-max Maximum_Value]
[-legal_value_list Value_List]
[-string_to_int_map Value_Map]
[-save_on_exit Boolean_Value]

Key             String
Category        String
ValueType       one of [bool|integer|double|color|string|rect|point|size]
Value           Depends on the value type
Description     String
Minimum_value   Minimum integer or double value
```

```
Maximum_value   Maximum integer or double value
Value_List      String List
Value_Map       A list of string pair lists, each mapping a string value
 to an integer value.
Boolean_Value   true | false
```

Arguments

`-key` *Key*

Key specifies the name of the key to create.

`-value_type` *ValueType*

ValueType specifies the data type of the value of the key. It must be one of [bool | integer | double | color | string].

`-value` *Value*

Value specifies the value of the key. The value should be set appropriately in accordance with its value type. The *bool* type accepts [true, false, 0, 1, on, off]. The *color* type accepts a named color, eg. "red" or a string specifying the color in this format "#RRGGBB", for example, "#0000FF".

`-category` *Category*

Category specifies the category that the key belongs to. If not specified, a default category will be assigned.

`-keep_value_if_exist`

This Boolean option specifies to keep current value if the key already exists. The default is that the key will be assigned the value given in the *-value* option.

`-read_only`

If given, this flag specifies that the preference key value is read-only and its value cannot be set after the key has been created.

`-description` *Description*

If given, the given description string with the meaning of the key is assigned to the preference key.

`-min` *Minimum_Value*

If given, specifies the minimum value for an integer or double type preference key.

`-max` *Maximum_Value*

If given, specifies the maximum value for an integer or double type preference key.

Feedback

`-legal_value_list` *`Value_List`*

If given, this option specifies discrete valid values for an iteger, double or string type preference key. Input the argument using Tcl list syntax: {{prefVal} ...}

`-string_to_int_map` *`Value_Map`*

If given, provides a map from a string key value to an integer value for preferences of integer type. Input the argument as a Tcl list of Tcl list where the internal list is a pair of the string value and the integer value: {{stringval intval} ...}

`-save_on_exit` *`Boolean_Value`*

If given, specifies whether the preference key value is saved to the user preference file upon existing the application. Acceptable argument is either *true* or *false*.

Description

This command creates a preference key with the specified key name. This key will have the specified value type and value, and it will be created under the specified category. If the category is not specified, the key will belong to the default category. Returns the value of the preference key.

Examples

The following example create a user-defined category *some_category* for storing preferences and then create an integer key *some_key* under that category with the initial value of 200.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type
 integer -value 200
```

See Also

- gui_create_pref_category
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category

Feedback

- gui_exist_pref_key
- gui_update_pref_file

---

gui_create_schematic

Creates a schematic view.

Syntax

string *gui_create_schematic*

```
[-clct objects]
[-size {w h}]
[-be_specific bool]
```

Data Types

```
objects  collection
{w h}    string
```

Arguments

`-clct objects`

Generates a schematic for the specified object collection. The current selection is used if not specified.

`-size {w h}`

Generates a schematic for the specified width (w) and height (h). Application default view size is used if not specified.

`-be_specific bool`

Controls the behavior of the schematic generation as to whether additional objects beyond those specifed using -clct. If not specifed, or specifed as false, additional objects will be included in the new schematic to provide connectivity and context. If specified as true, only the objects specified will be included in the new schematic.

Description

The command generates a new schematic view from the current selection or an object collection if specified.

Examples

The following example generates a schematic view of the hierarchical cell named RAM_1.

```
prompt> gui_create_schematic -clct [find cell RAM_1]
```

Feedback

See Also

- change_selection

---

gui_create_task

Create a task.

Syntax

gui_create_task

```
-task          TaskName
-item_root     ItemRootName
[-default]

TaskName          String
ItemRootName      String
```

Arguments

`-task` *TaskName*

TaskName specifies the name of the new task. Task name must be unqiue.

`-item_root` *ItemRootName*

ItemRootName specifies the name of the task root that will be associated with the task. Item root names are unique. If the same item root is associated with multiple tasks, then the tasks share the same item root.

`-default`

This option specifies that this command will be the default task set as the current task. If another task is later created with this option, the the later setting overrides any previous settings.

Description

This command creates a task. A task defines the user task context for the whole application. Tasks may be used to configure menus and also to present a hierarchical list of task items to the user to navigate in the Task Assistant. The hierarhical list of task items is defined with the gui_create_task_item command.

Examples

The following simple example creates a new task, then defines a Tk widget task page, and an HTML task page, then creates task items referencing the task pages.

```
gui_create_task -task MyAppTask -item_root MyAppTaskRoot
```

Feedback

```
gui_create_task_page -name InputFormPage -format tk -source
 create_widget_proc
gui_create_task_page -name AppDocument -format html
 -source ./MyAppDocument.html

gui_create_task_item -name "Application->Input Form" -task MyAppTask
 -page InputFormPage
gui_create_task_item -name "Application->Documentation" -task MyAppTask
 -page AppDocument
```

See Also

- gui_create_task_item
- gui_create_task_page

---

gui_create_task_item

Create an item in the task tree to access in the Task Assistant.

Syntax

gui_create_task_item

```
-name           ItemName
<-task TaskName | -item_root ItemRootName>
-page           PageName
[-search_terms  TermsList]

ItemName          String
TaskName          String
ItemRootName      String
PageName          String
TermsList         List
```

Arguments

`-name ItemName`

ItemName specifies the hierarchical name given to the task item being created. This is also the text displayed to the user in the Task Assistant.

`-task TaskName`

TaskName specifies the task in which this task item is created. If a task name is used to create the task item, the item is inserted into the task item root associated with the task in the create_task command. If other tasks use the same item root, they will also see the new task item created with this command. This option is mutually exclusive with the -item_root option. One must be specified but not both.

Feedback

`-item_root` *`ItemRootName`*

ItemRootName specifies the task item root in which this task item is created. Item root names are associated with task names with the gui_create_task command and allows multiple tasks to share a single task item root. This option is mutually exclusive with the -item_root option. One must be specified but not both.

`-page` *`PageName`*

PageName specifies the page content factory associated with this task item. When the user selects the task item, the page specified by this option will be constructed and shown in the Task Assistant.

`-search_terms` *`TermsList`*

TermsList specifies additional terms that can be matched by the command search feature to find this task item. For example, if the task page allows the user to specify values for input parameters to a command, you may want to associate the name of that command as additional search term for the task item.

Description

This command creates a task item in the given task. Task items are defined in a hierarchy with hierarchy levels delimited by the separator string "->". The hierarchy of task items are shown in the task item navigation tree in the Task Assistant.

When the user selects a leaf item in the task navigation tree, the Task Assistant will show the task page associated with the item by this command. Currently, the pages can either be constructed from an HTML document or from a Tk widget factory.

Examples

The following simple example adds a new task, then defines a Tk widget task page, and an HTML task page, then creates task items referencing the task pages.

```
gui_create_task -name MyAppTask -item_root MyAppTaskRoot

gui_create_tk_task_page -name InputFormPage -create_command
 create_widget_proc
gui_create_html_task_page -name AppDocument -path ./MyAppDocument.html

gui_create_task_item -name "Application->Input Form" -task MyAppTask
 -page InputFormPage
gui_create_task_item -name "Application->Documentation" -task MyAppTask
 -page AppDocument
```

See Also

- gui_create_task

Feedback

gui_create_task_page

Creates a task page with Command Form, HTML content or Tk Command Form.

Syntax

gui_create_task_page

```
-name taskPageName
-format pageType
-source contentSource
```

Data Types

taskPageName string

pageType string

contentSource string

Arguments

-name *taskPageName*

taskPageName specifies the name of a new task page.

-format *pageType*

Define this new page as a specified Type Page. *pageType* can be command, html, tk or tab. Which specifies the Command Form Page, HTML Page, Tk Command Form Page or Tabs Page.

-source *contentSource*

Define the content to be shown in this page. If this page is a Command Form Page or a Tk Command Form Page, *contentSource* is the command name. If it is a HTML Page, *contentSource* specifies the full path to the file with HTML source for the task page. If it is a Tabs Page, *contentSource* specifies a list of existing page names or page&title pairs.

Description

The *gui_create_task_page* command creates several kinds of task pages which can be shown in the task assistant. The name option value is the unique name of the page. The format option defines which kind of page to be created and the source option defines what content to be shown in the task page.

Examples

The following simple example creates three new task pages, then show them in the task assistant.

Feedback

```
prompt> gui_create_task_page -name page1 -format command -source
 gui_report_map
prompt> gui_create_task_page -name page2 -format html
 -source ./MyAppDocument.html
prompt> proc create_widget_proc {widgetName parentName} {
          p#populate $wigetName with Tk
        p}
prompt> gui_create_task_page -name page3 -format tk -source
 create_widget_proc
prompt> gui_create_task_page -name page4 -format tab -source {page1 page2
 {page3 title} }
prompt> gui_create_task -task "Testing Task" -item_root
 test_task_item_root
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 1"
 -page page1
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 2"
 -page page2
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 3"
 -page page3
prompt> gui_create_task_item -task "Testing Task" -name "Testing Page 4"
 -page page4
prompt> gui_show_task -task "Testing Task:Testing Page"
```

See Also

- gui_create_task
- gui_create_task_item

---

gui_create_timing_plot

Creates a plot for timing.

Syntax

status *gui_create_timing_plot*

```
[-type type]
-objects objects
-attribute attribute_name
-tcl_cmd tcl_command
[-view view_name]
[-title title_label]
[-num_bins num_bins]
[-min min_value]
[-max max_value]
[-greater_than gt_value]
[-less_than lt_value]
[-midpoint mid_value]
[-worst worst_side]
```

```
[-xlabel x_axis_label]
[-ylabel y_axis_label]
[-bin_width bin_width]
[-max_num_bins max_num_bins]
[-output_file file_path]
[-verbose]
```

Data Types

```
type                string
objects             list | collection | string
attribute_name      string
tcl_command         string
plot_name           string
view_name           string
title_label         string
num_bins            integer
min_value           float
max_value           float
gt_value            float
lt_value            float
mid_value           float
worst_side          string
x_axis_label        string
y_axis_label        string
bin_width           float
max_num_bins        integer
```

Arguments

`-type type`

Specifies the type of the timing plot. It currently supports *histogram*. The default value is *histogram*.

`-objects objects`

Specifies the objects to collect the data. It can be an object list, a collection, or a pre-defined category or objects, such as *failing_timing_paths*.

failing_timing_paths is identical to calling *get_timing_paths -delay_type min_max -pba_mode path -slack_lesser_than 0.0 -max_paths 2000000*.

`-attribute attribute_name`

Specifies the name of the attribute to construct the timing plot. The value of the named attribute will then be used. If an object does not have the attribute value, a message is issued and the object is excluded from the timing plot. The options *-attribute* and *-tcl_cmd* mutually exclusive.

If the object in *objects* is a timing path, a pre-defined word can be used as *attribute_name*, such as *slack*, *datapath_delay*, and *logic_depth*.

`-tcl_cmd` *`tcl_command`*

Specifies the Tcl command to construct the timing plot. The Tcl command is evaluated for each object in *objects*. If the Tcl command evaluation fails for any object, a message is issued and the object is excluded from the timing plot. The options *-attribute* and *-tcl_cmd* mutually exclusive.

The Tcl command should be a Tcl procedure with only one argument. The command takes an object in *objects* as the input and returns a real value as the output.

`-view` *`view_name`*

Specifies the name of a new or an existing view.

`-title` *`title_label`*

Specifies the label to use for the timing plot. If not specified, a default title is constructed by concatenating *y_axis_label* and *x_axis_label*.

`-num_bins` *`num_bins`*

Specifies the number of bins to use in creating the histogram. The value must be greater than or equal to 1.

`-min` *`min_value`*

Specifies the minimum value to use in the timing plot. The default value is negative infinity. Any object whose associated value is less than this value is to be dropped from the timing plot with an information message.

`-max` *`max_value`*

Specifies the maximum value to use in the timing plot. The default value is positive infinity Any object whose associated value is greater than this value is to be dropped from the timing plot with an information message.

`-greater_than` *`gt_value`*

Specifies the non-inclusive least value limit to use in the timing plot. The default value is negative infinity. Any object whose associated value is less than or equal to this value is to be dropped from the histogram with an information message.

`-less_than` *`lt_value`*

Specifies the non-inclusive greatest value limit to use in the timing plot.
The default value is positive infinity. Any object whose associated value is greater than or equal to this value is to be dropped from the histogram with an information message.

`-midpoint` *`mid_value`*

Specifies the midpoint value to use in interpreting histogram bins. The default value is zero if *objects* is *failing_timing_paths*. An x-axis tick mark is placed on the midpoint value and all bins to the *worst* side of the midpoint are colored red while all bins to the other side of the midpoint are colored green. The *worst* side is *none* by default but can be set to *left* or *right* using the *-worst* option.

`-worst` *`worst_side`*

Specifies whether values less than or greater than the midpoint are to be considered "worst." Allowed values are *left*, meaning that values less than midpoint are worse; *right*, meaning that values greater than the midpoint are worse. If *objects* is *failing_timing_paths*, the default value is *left*. If not the default value is *none* meaning neither side is worst.

`-xlabel` *`x_axis_label`*

Specifies the label to place along the x-axis of the timing plot. If *attribute_name* is specified, the default value is *attribute_name*. If *tcl_command* is specified, the default value is *tcl_command*.

`-ylabel` *`y_axis_label`*

Specifies the label to place along the y-axis of the timing plot. If *objects* is *failing_timing_paths*, the default value is *timing paths*. Otherwise, The default value is "Count".

`-bin_width` *`bin_width`*

Specifies the width of each bin. The value must be greater or equal to 1e-12.

`-max_num_bins` *`max_num_bins`*

Specifies the maximum number of bins allowed when the width of each bin is specified. The value must be greater than or equal to 1.

`-output_file` *`file_path`*

Specifies the file path of the output image. It supports different image format, including bmp, jpg, png, pbm, pgm, ppm, xbm, xpm, and svg.

It is useful to view the plot when the GUI is off.

`-verbose`

Outputs more information messages.

Description

The *gui_create_timing_plot* command constructs a timing plot with the values from the object in *objects* evaluated by the given attribute name or Tcl command.

Feedback

Examples

These examples show creation of custom histograms using the *gui_create_timing_plot* command.

The following example shows creating a timing histogram showing the slacks of the failing timing paths.

```
pt_shell> gui_create_timing_plot -type histogram \\
                -objects failing_timing_paths -attribute slack
1
```

If the GUI is off, the -output_file option should be specified. Otherwise, an error message will be issued.

```
pt_shell> gui_stop -close
pt_shell> gui_create_timing_plot -type histogram \\
                -objects failing_timing_paths -attribute slack
Error: The GUI is off and the -output_file option is not specified.
0
```

The following example shows creating a timing histogram using the *-attribute* flag to retrieve an attribute value for each object in a pin list. In this example, the *fanout_load* attribute value is used to bin all pins in the design. The plot is labeled with the title string "Pin Fanout" using the *-title* option. The new view is named *My View* by the *-view* option,

```
pt_shell> gui_create_timing_plot -type histogram \\
                -objects [get_pins *] -attribute fanout_load \\
                -title "Pin Fanout" -view "My View"
1
```

The following example shows creation of another histogram binning cells by area. The plot is added into the view named *My View* in the previous example.

```
pt_shell> gui_create_timing_plot -type histogram \\
                -objects [get_cells *] -attribute area \\
                -title "Cell area" -view "My View"
1
```

The following example shows creation of a histograms using the *-tcl_cmd* flag to use a user-defined Tcl procedure to retrieve a value for each object in *objects*. First, we define a Tcl procedure named "myproc" that retrieves the *fanout_load* attribute value for the object that is passed in the procedure argument.

```
proc myproc { obj } {
    set result [get_attribute $obj fanout_load]
    return $result
}
```

The procedure "myproc" can then be used in the *gui_create_timing_plot* command.

Feedback

```
pt_shell> gui_create_timing_plot -type histogram -tcl_cmd myproc \\
          -title "Pin Fanout" -objects [get_pins *]
1
```

In this example, the embedded command *[get_pins *]* is first interpreted to create a collection of all pins in the design. The Tcl procedure "myproc" is then invoked for each pin object in the collection to retrieve a value per pin to use in creating the histogram.

See Also

- collections
- get_timing_paths

---

gui_create_tk_palette_type

Create a type of Tk palette.

Syntax

```
status gui_create_tk_palette_type
-type string
-title string
[ -icon string ]
-window_types string_list
[ -dock_edge dock_edge ]
-create_command string
```

Data Types

```
dock_edge  string
```

Arguments

`-type string`

Unique palette type name for the palette.

The type name must be non-empty, must start with a letter or underscore, and must only contain letter, number or underscore characters.

`-title string`

Title string for the palette.

`-icon string`

Icon to display for the palette.

`-window_types string_list`

List of window types which the palette is to be added to.

Feedback

`-dock_edge` *dock_edge*

The default edge to add the palette

```
left   - dock on left edge
right  - dock on right edge
top    - dock on top edge
bottom - dock on bottom edge
```

`-create_command` *string*

The option specifies the tcl command to be run when the palette is created so that the palette's tk widget can be populated with tk child widgets and any other initialization can be performed.

The tcl procedure takes two arguments 'windowName' and 'parentName' where 'windowName' is the tk palette widget to populate and 'parentName' is the name of the parent window.

Description

Creates a user defined tk palette when a window of any of the specified types is created.

Examples

The following example creates a palette of type 'mypalette', with a title of 'My Palette' which calls the 'build_my_palette' tcl procedure to populate the palette with tk widgets whenever a new window of type 'LayoutWindow' is created.

The 'build_my_palette' creates a text entry filed in a frame for the palette.

```
shell> proc build_my_palette {windowName parentName} {
   set top_frame [frame $windowName.frame]
   pack $top_frame -fill both -expand 1
   set entry [entry $top_frame.entry -textvariable $::var]
   pack $entry
 }
shell> gui_create_tk_palette_type -type mypalette -title "My Palette" \\
  -create_command build_my_palette -window_types "LayoutWindow"
 -dock_edge right
```

---

gui_create_toolbar

Create a toolbar with the specified name.

Syntax

string *gui_create_toolbar* -name *ToolBarName*

```
[-title       Title]
[-dock_side   DockSide]
```

Feedback

```
[-hidden]
[-window_type WindowTypeName]

ToolBarName     String
Title           String
DockSide        String
WindowTypeName  String
```

Arguments

`-name ToolBarName`

ToolBarName is the toolbar identifier for the new toolbar.

`-title Title`

Title specifies the title or caption of the toolbar.

`-dock_side DockSide`

DockSide specifies the side of the window the tool docks to by default. The legal values are top, bottom, left, right.

`-hidden`

-hidden specifies the toolbar should be hidden by default.

`-window_type WindowTypeName`

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

Description

This command creates a toolbar with the given name and title. Returns the name of the toolbar created.

Examples

Create a toolbar called mytoolbar and add a button to it that invokes the File->Quit menu item.

```
gui_create_toolbar -name mytoolbar
gui_create_toolbar_item -name mytoolbar -menu "File->Quit"
```

See Also

- gui_set_hotkey
- gui_report_hotkeys

Feedback

gui_create_toolbar_item

Create a button in the specified toolbar.

Syntax

string *gui_create_toolbar_item* -toolbar *ToolBarName*

```
<-menu MenuName|-separator SeparatorName>
[-window_type WindowTypeName]
[-dropdown_menu MenuName]
[-dropdown_icon IconName]

ToolBarName     String
MenuName        String
SeparatorName   String
WindowTypeName  String
IconName        String
```

Arguments

`-toolbar ToolBarName`

ToolBarName specifies the toolbar that the new toolbar item will be added to.

`-menu MenuName`

MenuName specifies the name of the menu item to be invoked from this toolbar button. The "name" of a menu is a string whcih specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do so. This option is mutually exclusive with the -separator option.

Feedback

`-separator` *`SeparatorName`*

SeparatorName specifies the unique separator name within the toolbar. This option is mutually exclusive with the -menu option.

`-window_type` *`WindowTypeName`*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

`-dropdown_menu` *`MenuName`*

MenuName specifies the associated menu to add as a dropdown menu for the item.

`-dropdown_icon` *`IconName`*

IconName specifies the icon to display on the dropdown menu button.

If not specified a standard menu will be used.

Description

This command adds a toolbar button to the specified toolbar using information described in the -menu option. The -menu option specifies the hierarchical menu name for some menu item. Thus, the toolbar icon becomes another interface for a menu item. Returns the item name if successful.

Examples

Create a toolbar called mytoolbar and add a button to it that invokes the File->Quit menu item.

```
gui_create_toolbar -name mytoolbar
gui_create_toolbar_item -name mytoolbar -menu "File->Quit"
```

Create a drop down menu button that has actions that run a command

```
gui_create_menu -menu "Item 1" -tcl_cmd "echo 1" -root "my_root"
gui_create_menu -menu "Item 2" -tcl_cmd "echo 2" -root "my_root"
gui_create_toolbar -name "MyToolbar" -title "My Toolbar" -dock_side top
gui_create_toolbar_item -toolbar MyToolbar -menu "Item 1" \
  -dropdown_menu "my_root" -dropdown_icon my_icon
```

See Also

- gui_set_hotkey
- gui_report_hotkeys

Feedback

---

gui_create_utable

Create a new UserTable.

Syntax

string *gui_create_utable*

```
-name              Table_Name
-parent            Table_Name
-columns           Column_List
-timing_path       Tim_Clct
[-expr_columns     Column_Expr_List]
[-meta_obj         MetaObj_Name]
[-replace]
[-fixed_name]

Table_Name         String
Column_List        List of Strings
Column_Expr_List   A list of column-expression pairs
Tim_Clct           Timing Path Collection
MetaObj_Name       String
```

Arguments

`-name` *Table_Name*

The name of the user table to create.

`-parent` *Table_Name*

The name of the parent table that is related to this user table.

`-columns` *Column_List*

A TCL list of column names for the table with optional data type Xpostfix information.

Feedback

`-expr_columns` *`Column_Expr_List`*

This allows you to create an expression that fills a table column with the result of that expression. The option is a list of column-expr pairs. Thus the list must have an even number of elements where the odd elements are the column names and the even number elements are the associated expression.The expression can reference other column values by surounding the column name in the following syntax: @{...}. The expression itself can include operators like '+','-','/' and '*' and parentheses. See the example below and see *gui_append_utable* for more info.

`-timing_path` *`Tim_Clct`*

Create and fill a Timing Point Table with the given Timing Path Collection. This is exclusive with the -columns option.

`-meta_obj` *`MetaObj_Name`*

Optional MetaData object name used to define table meta data. See the command gui_set_utable_meta for more information.

`-replace`

If a table already exists with that name, replace it.

`-fixed_name`

Mark the table name as 'fixed' so that it can not be changed in the GUI.

Description

```
This command creates a new user table with the given list of column
 names.
In addition, column names can have postfixes that describe the data type
 of that column.
Or one can use a MetaData object to define the column data types.

Important: This command returns the actual table name created which might
be different if the given table name had bad characters or already
 exists.
So, capture the return value and use it in future calls to this table.

Supported column name postfixes are:


  :string
  :int
  :double
  :cell
  :net
  :port
  :pin
  :point
```

:endpoint
:file

Examples

```
The following example creates a new UserTable with the given list of
 columns. The last
three columns also have some data type postfixes. Also note that the
 return value of
the create is captured since the name could be altered if it contained
 illegal chars.
This captured name is then used in following commands like the append
 command.
This example also adds an empty column at the end which is then filled
 with an
expression that references other columns.

set tableName [gui_create_utable -name MyTable -replace \\
                    -columns {
                      full_name
                      arrival:double
                      delay:double
                      {Expr Col:double}
                    } \\
                    -expr_columns {
                      {Expr Col} { 100 * (@{arrival} + @{delay}) }
                    }
               ]

# Append a TCL row: (note: that expr column is empty: {} )
gui_append_utable -name $tableName -rows { {my/path 2.2 3.3 {} } }
```

See Also

- gui_append_utable
- gui_change_selection_utable
- gui_close_utable
- gui_export_utable
- gui_get_utable
- gui_import_utable
- gui_set_utable
- gui_set_utable_meta
- gui_open_utable

Feedback

- gui_show_utable
- gui_write_utable

---

gui_create_vm

Creates a new visual mode container for visual mode buckets. The buckets contain coloring attributes for a collections of design objects.

Syntax

string *gui_create_vm*

```
-name identifier
[-update_cmd update_command]
[-title label]
[-help_topic subject]
[-infotip infotip]
[-discrete  true | false]
[-icon string]
[-netfilter string]
[-float bool]
[-top_exaggeration float]
[-mid_exaggeration float]
[-bot_exaggeration float]
[-show_only_pins_of_nets bool]
[-enable_bucket_delete bool]
```

Data Types

```
identifier          string
update_command      string
label               string
subject             string
infotip             string
net_filter          string
icon_spec           string
```

Arguments

`-name identifier`

Specifies the name that identifies the visual mode. Use this name when another visual mode command requires a visual mode name.

`-update_cmd update_command`

Specifies a Tcl command to execute when the visual mode is accessed. Use this option to update the state of the visual mode if its contents are likely to change under certain circumstances.

`-title` *label*

Specifies an optional title that is used to label the visual mode in the GUI. The *label* string can contain space characters. By default, the title is the empty string, and the *-name* option value is used to provide the label.

`-help_topic` *subject*

Specifies the help topic text string. This text is used as the subdirectory to find a help page related to the visual mode.

`-infotip` *infotip*

Specifies the infoTip text string.

`-discrete true | false`

Specifies an optional true or false string indicating whether the buckets of this visual mode are discrete or continuous. Only discrete buckets can be reordered. By default, visual modes are discrete.

`-icon` *string*

Specifies the icon file. The file contains an icon image for the GUI menus.

`-netfilter` *string*

Specifies the net connection filtering.

`-float true | false`

Specifies whether the bucket contents have a floating point range. By default, this option is false.

`-top_exaggeration` *float*

Specifies the exaggeration for the top bucket. The value should be greater than or equal to zero. By default, the value is 0.

`-mid_exaggeration` *float*

Specifies the exaggeration for the middle bucket. The value should be greater than or equal to -1. By default, the value is -1.

`-bot_exaggeration` *float*

Specifies the exaggeration for the bottom bucket. The value should be greater than or equal to zero. By default, the value is 0.

`-show_only_pins_of_nets true | false`

Specifies whether the layout shows pins of net objects in buckets. By default, the value is false.

Feedback

`-enable_bucket_delete true | false`

Specifies whether the buckets can be deleted from context menu. Only discrete visual mode buckets can be deleted. By default, the value is false.

Description

The *gui_create_vm* command creates an instance of a visual mode. The visual mode provides a container for visual mode buckets, which associate coloring attributes with collections of design objects.

If the -update_cmd that you specify needs input from the user, then you can define the arguments to your procedure with define_proc_attributes, and then use the gui_show_command_form command for your procedure as the -update_cmd. This will show a form that will prompt for the arguments to your procedure and then it will call your proc to update the visual mode using those values. See the man pages for gui_show_command_form and define_proc_attributes for additional details.

Examples

The following example creates a new visual mode named mycoloring1 with the GUI label "Cells Colored By X":

```
prompt> gui_create_vm -name mycoloring1 -title {Cells Colored By X}
```

See Also

- gui_create_vmbucket
- gui_get_vm
- gui_get_vmbucket
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vm
- gui_set_vmbucket
- gui_update_vm
- gui_show_command_form
- define_proc_attributes

---

gui_create_vm_objects

Create objects to hold annotations in visual modes

Feedback

Syntax

string *gui_create_vm_objects <clct>*

```
string <clct>
```

Arguments

<clct>

Objects to convert.

Description

Layout visual modes are used to color objects in the layout given certain criteria. Sometimes you want not only object coloring but some additional annotations displayed with the objects in a bucket.

In order to create this type of visual mode, you need special type of object that can hold annotations. Use this command to create these special objects.

Objects of this type support name and core_obj attribute. The core_obj attribute returns the object that was used to create this object.

Examples

shell> *set objs [gui_create_vm_objects $cells]*

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_update_vm_annotations

---

gui_create_vmbucket

Create new Visual Mode Bucket

Syntax

string *gui_create_vmbucket* -vmname *mode_identifier*

-name *bucket_identifier* [-infotip *infotip*] [-netfilter *net_filter*] [-color *color*] [-pattern *pattern*] [-exaggeration double] [-number int] [-maxval double] [-minval double] [-visible *visibility*] [-title *label*] [-collection *handle*] [-above *bucket_identifier* | -below *bucket_identifier* -at *top*| *bottom*]

```
string mode_identifier
string bucket_identifier
string infotip
```

Feedback

```
string net_filter
string color
string pattern
string visibility
string label
string handle
string bucket_identifier
string bucket_identifier
string top|bottom
```

Arguments

`-vmname mode_identifier`

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

`-name bucket_identifier`

Specifies the name by which the visual mode bucket will be identified. It is used as the argument when associated visual mode commands require a visual mode bucket name to be specified.

`-infotip infotip`

String for infotip.

`-netfilter net_filter`

Specifies net connection filtering.

`-color color`

Specifies bucket rendering color. Supports color naming by RGB triplet (for example: "#FFFF00" for yellow) as well as standard names (for example: "yellow").

`-pattern pattern`

Specifies bucket rendering fill pattern. Supported patterns are: SolidPattern, HorPattern, VerPattern, CrossPattern, BDiagPattern, FDiagPattern, DiagCrossPattern, Dense1Pattern, Dense2Pattern, Dense3Pattern, Dense4Pattern, Dense5Pattern, Dense6Pattern, Dense7Pattern, NoBrush.

`-exaggeration double`

Specifies bucket min pixel exaggeration value >= 0.

`-number int`

Specifies bucket display number value >= 0 that is used to provide a display number for the visual mode bucket in the gui. If the -number option is not specified, it will default to the number of objects in the bucket.

Feedback

`-maxval double`

Specifies bucket maximum value.

`-minval double`

Specifies bucket minimum value.

`-visible` *visibility*

Specifies initial visibility of bucket. Valid values are TRUE and FALSE.

`-title` *label*

Specifies an optional string (that can include embedded spaces) that is used to provide a label for the visual mode bucket in the gui. If the -title option is not specified, it will default to an empty string and the -name argument value value will be used to provide the labeling instead.

`-collection` *handle*

Tcl object collection to be colored by this visual mode bucket.

`-above` *bucket_identifier*

Specifies an optional visual mode bucket name above which the current bucket is to be rendered.

`-below` *bucket_identifier*

Specifies an optional visual mode bucket name below which the current bucket is to be rendered.

`-at` *top|bottom*

Specifies an optional string indicating whether the current visual mode bucket is to be rendered at the top or bottom of all other buckets. Valid values are top and bottom.

Description

The gui_create_vmbucket command creates an instance of a visual mode bucket. The visual mode bucket is a member of a specific visual mode. The visual mode bucket can be assigned coloring styles and a collection of design objects to which the coloring will be applied when they are rendered as a part of the visual mode.

Examples

Create a visual mode with two buckets, one whose objects will be colored red, and another whose objects will be colored blue:

shell> *gui_create_vm -name foo1 -title "My Visual Mode"*

shell> *gui_create_vmbucket -vmname foo1 -name red -title "Red" -color red*

Feedback

shell> *gui_create_vmbucket -vmname foo1 -name blue -title "Blue" -color blue -below red*

See Also

- gui_create_vm
- gui_get_vm
- gui_get_vmbucket
- gui_list_vm
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vm
- gui_set_vmbucket
- gui_update_vm

---

gui_create_window

Creates a window of the specified window type.

Syntax

```
string gui_create_window
-type window_type
[ -parent parent_window_id ]
[ -show_state window_state ]
[ -title title ]
[ -rect rect | -size {width height} ]
[ -icon icon ]
```

Data Types

```
window_type          string
parent_window_id     string
window_state         string
title                string
rect                 rectangle
width                integer
height               integer
icon                 string
```

Feedback

Arguments

`-type` *`window_type`*

Specifies the type of window you are creating. The tool creates an instance of this window type.

You can display a list of the supported window types by using the *gui_get_window_types* command.

`-parent` *`parent_window_id`*

Specifies the window ID of the parent top-level window for the view window you are creating.

If you do not specify a window ID and the window type is not a top-level window type, the tool uses the current top-level window. If there is no current top-level window, the tool creates a new top-level window automatically to host the window you are creating.

`-show_state` *`create_window_state`*

Specifies the window state in which to display the window you are creating. The result varies depending on whether the window is a top-level window or a view window.

For a top-level window, you can specify one of the following states:

```
maximized -- show maximized (fill whole screen)
minimized -- show iconified
normal    -- show at preferred size
```

Note that the window manager might not fully honor the specified state. See your window manager documentation for more details.

For a view window, you can specify one of the following states:

```
maximized -- show maximized in view area and raise to top
minimized -- show iconified in view area
normal    -- show at preferred size in view area and raise to top
```

If you display a view window in its maximized state, the tool also maximizes any existing view windows. If you display a view window in its minimized or normal state, the tool changes any maximized view windows to their normal states.

The default is -show_state normal.

`-title` *`title`*

Specifies the title for the top-level or view window you are creating.

Note that the window manager might not fully honor the specified top-level window title string. See your window manager documentation for more details.

Feedback

`-rect rect`

Specifies the size and position of the window in the following format:

`{{x1 y1} {x2 y2}}`

The coordinates {*x1 y1*} specify the location of the top-left corner, and the coordinates {*x2 y2*} specify the location of the bottom-right corner. These coordinates are relative to the top-left corner of the screen, for a top-level window, or to the top-right corner of the view area in the parent top-level window, for a view window.

Note that the window manager might not fully honor the specified top-level window geometry. See your window manager documentation for more details.

The *-rect* option and the *-size* option are mutually exclusive.

`-size {width height}`

Specifies the width and height of the window.

Note that the window manager might not fully honor the specified top-level window geometry. See your window manager documentation for more details.

The *-size* option and the *-rect* option are mutually exclusive.

`-icon icon`

Specifies the image to be used for the top-level or view window icon.

For a view window, the icon appears in the top-left corner of the title bar when the window is in its normal or minimized state and at the left side of the menu bar when the window is maximized.

For a top-level window, the icon might not be displayed when the window is in its normal, minimized, or maximized state. See your window manager documentation for more details.

Description

This command creates a window of the specified type and returns the window ID. The window type that you specify controls whether the window is a top-level window or a view window.

Examples

The following example creates a top-level layout window and a minimized layout view window.

```
prompt> set top [gui_create_window -type LayoutWindow]
LayoutWindow.1
prompt> gui_create_window -type Layout -parent $top \\
```

Feedback

```
   -show_state minimized
layout.1
```

See Also

- gui_close_window
- gui_exist_window
- gui_show_window
- gui_get_window_types
- gui_get_window_ids
- gui_set_active_window
- gui_get_current_window

---

gui_create_window_toolbar_group

Create group in view toolbar

Syntax

```
status gui_create_window_toolbar_group
[ -window window_id ]
[ -toolbar string ]
[ -group string ]
-type string
-name string
[ -label string ]
[ -icon string ]
[ body ]
```

Data Types

window_id string

Arguments

-window *window_id*

Specifies the view name id.

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

-toolbar *string*

Specifies the toolbar id.

Feedback

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

`-group` *string*

Specifies the parent group name.

`-type` *string*

Specifies the group type.

If the string '?' is specified for the type, the command will return a list of allowed types.

`-name` *string*

Specifies the unique name for the group.

Note: The command will fail if the name has already been used in a toolbar.

`-label` *string*

Specifies the optional label for the group.

`-icon` *string*

Specifies the optional icon for the group.

body

Tcl commands to add items or sub groups to the group.

Description

This command adds a new control group to the specified view's toolbar.

Examples

The following example will create buttons in a stamp group in the toolbar of the specified Charts View and attach a callback when any button is pressed.

```
shell> proc buttonPressed { args } { echo "Button Pressed" }
shell> proc init_my_toolbar { args } {
shell>   gui_create_window_toolbar_group -name buttons -type stamp -label
 Controls {
shell>     set button1 [gui_create_window_toolbar_item -name button1
 -type button]
shell>     set button2 [gui_create_window_toolbar_item -name button2
 -type button]
shell>     set_attribute $button1 label "Button 1"
shell>     set_attribute $button2 label "Button 2"
shell>     set_attribute $button1 activated_proc buttonPressed
shell>     set_attribute $button2 activated_proc buttonPressed
shell>   }
shell> }
```

Feedback

```
shell> gui_create_window_toolbar_type -type my_toolbar -name "My Toolbar"
 \\
shell>   -view_types Charts -command init_my_toolbar
shell> set top [gui_create_window -type TopLevel]
shell> set charts [gui_create_window -type Charts -parent $top]
shell> gui_show_window_toolbar -window $charts
```

See Also

- gui_create_window_toolbar_type
- gui_create_window_toolbar_item
- gui_create_window
- gui_show_window_toolbar
- get_attribute

---

gui_create_window_toolbar_item

Create item in view toolbar

Syntax

```
status gui_create_window_toolbar_item
[ -window window_id ]
[ -toolbar string ]
[ -group string ]
-type string
-name string
[ -label string ]
[ -icon string ]
```

Data Types

```
window_id  string
```

Arguments

`-window window_id`

Specifies the view name id.

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

`-toolbar string`

Specifies the toolbar id.

Feedback

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

`-group` *string*

Specifies the optional name of the group to add the item to.

Note: Only allowed when no active group.

`-type` *string*

Specifies the item type.

If the string '?' is specified for the type, the command will return a list of allowed types.

`-name` *string*

Specifies the unique name for the item.

Note: The command will fail if the name has already been used in a toolbar.

`-label` *string*

Specifies the label for the item.

`-icon` *string*

Specifies the icon for the item.

Description

This command adds a new control item to the specified view's toolbar.

The control is added to the current active group or in it's own stamp if there is no current group

Examples

The following example will create a button in the toolbar of the specified Charts View and attach a callback when the button is pressed.

```
shell> proc buttonPressed { args } { echo "Button Pressed" }
shell> proc init_my_toolbar { args } {
shell>   set button [gui_create_window_toolbar_item -name button -type
 button]
shell>   set_attribute $button activated_proc buttonPressed
shell> }
shell> gui_create_window_toolbar_type -type my_toolbar -name "My Toolbar"
 \\
shell>   -view_types Charts -command init_my_toolbar
shell> set top [gui_create_window -type TopLevel]
shell> set charts [gui_create_window -type Charts -parent $top]
shell> gui_show_window_toolbar -window $charts
```

Feedback

See Also

- gui_create_window_toolbar_type
- gui_create_window_toolbar_group
- gui_create_window
- gui_show_window_toolbar
- get_attribute

---

gui_create_window_toolbar_type

Create new type for view toolbar

Syntax

```
status gui_create_window_toolbar_type
-type string
[ -name string ]
[ -icon string ]
-view_types string_list
-command string
[ -apply ]
```

Arguments

`-type string`

Specifies the toolbar type.

The toolbar type must be unique.

`-name string`

Specifies the user visible name for the toolbar.

If not specified then the type name is used.

`-icon string`

Specifies the optional icon for the toolbar.

`-view_types string_list`

Specifies the window view types which the toolbar can be added to.

`-command string`

Specifies the command to run to initialize the toolbar.

`-apply`

Specifies that the type will be applied to existing views.

Feedback

Description

This command defines a new toolbar type which will be populated using the user supplied tcl proc when the toolbar is created in the associated view.

Examples

The following example will create a user toolbar in the Charts view with a button.

```
shell> proc buttonPressed { args } { echo "Button Pressed" }
shell> proc init_my_toolbar { args } {
shell>   gui_create_window_toolbar_group -name stamp -type stamp {
shell>     set button [gui_create_window_toolbar_item -name button -type
 button]
shell>     set_attribute $button label "Button"
shell>     set_attribute $button activated_proc buttonPressed
shell>   }
shell> }
shell> gui_create_window_toolbar_type -type my_toolbar -name "My Toolbar"
 \\
shell>   -view_types Charts -command init_my_toolbar
shell> set window [gui_create_window -type TopLevel]
shell> set charts [gui_create_window -type Charts]
shell> gui_show_window_toolbar -window $charts
```

See Also

- gui_create_window_toolbar_group
- gui_create_window_toolbar_item
- gui_create_window
- get_attribute

---

gui_define_charts_proc

Define tcl command to use in charts expression evaluation

Syntax

```
status gui_define_charts_proc
name
args
body
```

Arguments

name

Name of the procedure.

Feedback

The procedure name must be non-empty, start with an underscore or a letter and contain only underscore or letter characters.

args

The arguments of the procedure.

body

The body of the procedure.

Description

Define custom tcl command to use in charts expression evaluation.

The syntax and arguments are the same as the standard tcl *proc* command.

If the arguments and body strings are empty then any existing procedure of that name is deleted.

To list all the defined charts procedures you can use the command:

```
shell> gui_get_charts_data -global -name procs
```

To get the args and body for an existing procedure you can use the command:

```
shell> gui_get_charts_data -global -name proc_data -data <proc_name>
```

Examples

The following example changes a model to create a new column with the square root of values in the first column.

```
shell> gui_define_charts_proc test_proc { arg } { return [expr
 {$arg/2.0}] }
shell> gui_change_charts_model -model $model -add -expr
 "test_proc(column(0))" -type real
```

See Also

- gui_change_charts_model
- gui_get_charts_data

---

gui_delete_attrgroup

Removes a group of attributes or all attribute groups for an object type.

Syntax

```
status gui_delete_attrgroup
-class design_object
-name name
```

Feedback

```
[-all]
[-quiet]
```

Data Types

```
design_object       string
name                string
```

Arguments

`-class design_object`

Specifies the type of design object.

`-name name`

Specifies the name of the attribute group to be removed for the specified object type.

`-all`

Removes all attribute groups defined for the specified object type.

`-quiet`

Keep quiet and don't print any messages if command was not successfull.

Description

The *gui_delete_attrgroup* command removes the specified attribute group or all attribute groups for the specified object type.

Examples

```
prompt> gui_delete_attrgroup -class Cell -name "Hierarchy"
```

```
prompt> gui_delete_attrgroup -class Cell -all
```

See Also

- gui_create_attrgroup
- gui_list_attrgroups
- gui_update_attrgroup

---

gui_delete_menu

Delete a menu item for the toplevel menubar or a context menu

Syntax

string *gui_delete_menu*

Feedback

```
[-menu HierMenuName | -all]
[-window_type WindowTypeName | -root ContextMenuRoot]

HierMenuName    String
WindowTypeName  String
ContextMenuRoot String
```

Arguments

`-menu HierMenuName`

HierMenuName specifies the hierarchical menu name of the menu item to be deleted. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item under the File menu. These names are allowed to also specify the mnemonic for the menu text which is specified with an embedded '&', but they are not required to do so. This option is mutually exclusive with the -all option.

`-all`

This option flag specifies all menu items in a menu root to be deleted. The option cannot be given with the -window_type option.

`-window_type WindowTypeName`

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

`-root ContextMenuRoot`

ContextMenuRoot specifies the context menu root to from which the menu item will be added. A context menu root name groups menu items to be included in a pop-up context menu. A context menu is usually brought up by right clicking on a window and is used to provide context-sensitive features. (Note the difference between a context menu root and the menu root use by a toplevel window's menu bar .)

Description

This command deletes a menu item for the menu bar of a top level window. The menu hierarchy separator is denoted by "->". If there is a toolbar button or hotkeys defined for the menu being deleted they will also be deleted. If the path to the menu is a submenu that menu item and all its sub-items will be deleted.

The option -all may be used with a menu root name to remove all menu items from a menu. This is useful if a user wishes to customize or modify an entire context menu.

These settings are stored either in a user-specific setup file or one that is shared across the installation. The gui initialization sequence will load builtin system configuration, and then apply the installation specific customizations, and finally the user's customizations.

The installation specific setup file is specified by the variable gui_custom_setup_file and will have a default value of $::synopsys/admin/setup/.synopsys_galileo_gui.tcl. This can be overridden in the .synopsys_dc.setup file if it is desired. If the file specified by this variable exists then it will be loaded after the gui is initialized.

The user setup file is ~/.synopsys_galileo_gui.tcl.

Examples

The following example deletes the Quit item from the File menu.

```
gui_delete_menu -menu "File->Quit"
```

See Also

- gui_set_hotkey
- gui_report_hotkeys
- gui_create_menu
- gui_create_toolbar
- gui_delete_toolbar
- gui_create_toolbar_item
- gui_delete_toolbar_item
- gui_default_window_type
- gui_custom_setup_files

---

gui_delete_toolbar

Delete a toolbar with the specified name.

Syntax

string *gui_delete_toolbar* -name *ToolBarName*

```
[-window_type WindowTypeName]

ToolBarName      String
WindowTypeName   String
```

Feedback

Arguments

`-name` *`ToolBarName`*

ToolBarName is the toolbar identifier for the toolbar to be deleted.

`-window_type` *`WindowTypeName`*

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

Description

This command deletes a toolbar and all of its toolbar items.

Examples

Delete a toolbar called mytoolbar.

```
gui_delete_toolbar -name mytoolbar
```

See Also

- gui_set_hotkey
- gui_report_hotkeys
- gui_create_menu
- gui_delete_menu
- gui_create_toolbar
- gui_create_toolbar_item
- gui_delete_toolbar_item
- gui_get_toolbar_names
- gui_default_window_type
- gui_custom_setup_files

---

gui_delete_toolbar_item

Remove a toolbar button specified by the menu name from the specified toolbar.

Feedback

Syntax

string *gui_delete_toolbar_item* -toolbar *ToolBarName*

```
<-menu MenuName|-separator SeparatorName>
[-window_type WindowTypeName]

ToolBarName      String
MenuName         String
SeparatorName    String
WindowTypeName   String
```

Arguments

`-toolbar ToolBarName`

ToolBarName specifies the toolbar to be modified.

`-menu MenuName`

ItemName specifies the hierarchical menu name for some menu item, that this toolbar item is associated with. This option is mutually exclusive with the -separator option.

`-separator SeparatorName`

SeparatorName specifies the unique separator name within the toolbar. This option is mutually exclusive with the -item option.

`-window_type WindowTypeName`

WindowTypeName specifies the type of top level window whose menu bar is to be modified. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

Description

This command deletes a toolbar item specified by *MenuName* or *SeparatorName* from the specified toolbar. Returns the name of the deleted toolbar item if successful.

Examples

Delete the button bound to File->Quit from the toolbar mytoolbar.

```
gui_delete_toolbar_item -name mytoolbar -menu "File->Quit"
```

See Also

- gui_set_hotkey
- gui_report_hotkeys

Feedback

- gui_create_menu
- gui_delete_menu
- gui_create_toolbar
- gui_delete_toolbar
- gui_create_toolbar_item
- gui_get_toolbar_names
- gui_default_window_type
- gui_custom_setup_files

---

gui_delete_window_toolbar_item

Delete view toolbar items

Syntax

```
status gui_delete_window_toolbar_item
[ -window window_id ]
[ -toolbar string ]
{ -item string | -all }
```

Data Types

```
window_id  string
```

Arguments

`-window window_id`

Specifies the view name id.

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

`-toolbar string`

Specifies the name of the toolbar to delete from.

`-item string`

Specifies the item name to delete.

`-all`

If set option is *not* specified then all items in the toolbar are deleted.

If set option is specified then all items in the set are deleted.

Feedback

Description

This command deletes one or more control items from the specified view's toolbar.

Examples

The following example will create a button in the toolbar of the specified Charts View and delete it.

```
shell> proc init_my_toolbar { args } {
shell>   set button [gui_create_window_toolbar_item -name button -type
 button]
shell> }
shell> gui_create_window_toolbar_type -type my_toolbar -name "My Toolbar"
 \\
shell>   -view_types Charts -command init_my_toolbar
shell> set window [gui_create_window -type TopLevel]
shell> set charts [gui_create_window -type Charts]
shell> gui_show_window_toolbar -window $charts
shell> gui_delete_window_toolbar_item -window $charts -toolbar my_toolbar
 -item button
```

See Also

- gui_create_window_toolbar_type
- gui_create_window_toolbar_item
- gui_create_window_toolbar_group
- gui_create_window
- gui_show_window_toolbar

---

gui_edit_vmbucket_contents

Edit the collection contents of a Visual Mode Bucket

Syntax

string *gui_edit_vmbucket_contents* -vmname *mode_identifier*

-name *bucket_identifier* [-add | -remove | -replace] [-collection *handle*]

```
string  mode_identifier
string  bucket_identifier
string  handle
```

Feedback

Arguments

`-vmname` *mode_identifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

`-name` *bucket_identifier*

Specifies the name of the visual mode bucket to be modified. To see the list of buckets associated with a specific visual mode use the gui_get_vm command with the -buckets option.

`-add`

Add the collection to the collection already in the bucket

`-remove`

Remove the collection from the collection already in the bucket

`-replace`

Replace the collection already in the bucket

`-collection` *handle*

Tcl object collection to be colored by this visual mode bucket.

Description

The gui_edit_vmbucket_contents command modifies the collection stored in a visual mode bucket. The collection can either be appended to, removed, or replaced. The flags to control this are mutually exclusive.

Examples

Add the current selection to the collection in the bucket 0 of the SNAPSHOT visual mode:

shell> *gui_edit_vmbucket_contents -vmname SNAPSHOT -name 0 -add -collection [get_selection]*

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm
- gui_get_vmbucket
- gui_list_vm

Feedback

- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vm
- gui_update_vm

gui_error_browser

Show or hide the Error Browser Dialog

Syntax

string *gui_error_browser*

[-hide | -show | -toggle]

Arguments

`-hide`

Hides the Error Browser Dialog.

`-show`

Shows the Error Browser Dialog. If no option flag is provided, this is the default.

`-toggle`

Show the Error Browser Dialog if it is currently hidden, else hide it.

Description

The *gui_error_browser* command controls the visibility of the Error Browser Dialog. The Error Browser Dialog displays violations with physical shapes recorded by a checker, for example a DRC engine. This command is valid only when there is a design open. If the Error Browser Dialog is shown and it was not previously shown. If the Error Browser Dialog is shown and it was previously shown, it will show the error data last loaded.

Examples

The following example creates and shows the Error Browser Dialog:

gui_error_browser -show

gui_eval_command

Executes and optionally logs a tool command language (Tcl) command.

Feedback

Syntax

string *gui_eval_command*

```
-command command
[-echo]
[-history]
[-honor_preview | -preview]
```

Data Types

command string

Arguments

`-command` *command*

Specifies the Tcl command to execute and to record in the command replay log.

`-echo`

Echoes the command and displays the command result in the console log view. By default, the tool does not echo the command or display the result in the log view.

`-history`

Appends the command to the command history list in the console history view. By default, the tool does not append the command to the command history list.

`-honor_preview`

This option is mutually exclusive with the -preview option. Honors the Preview Command Only settting in the dialog. By default, the tool does not honor the Preview Command Only settting.

`-preview`

This option is mutually exclusive with the -honor_preview option. Runs the given command in preview mode, irregardless of the current preview mode set from a command dialog. When -preview is given, -echo is turned on since the previewed command cannot be seen in the xterm if -echo is turned off. As with all commands issued from command dialogs when in preview mode, if a script editor is present, then the previewed command is redirected to the scripte editor. Using this option does not set or clear the global preview mode in the application.

Description

This command executes a Tcl command and records it in the command replay log. Use the *-echo* option to echo the command and its result in the console log view Use the *-history* option to append the command to the command history list. Use the

Feedback

-honor_preview option to honor the Preview Command Only settting. Use the *-preview* option to preview the command only without executing it.

The result of a nested command is passed back as the result of *gui_eval_command*.

Examples

The following example evaluates the command "echo abc".

```
prompt> gui_eval_command -command {echo abc}
abc
```

The following example previews the command "echo abc".

```
prompt> gui_eval_command -command {echo abc} -preview
abc
```

---

gui_eval_task_command

Executes a command from a task assistant page

Syntax

string *gui_eval_task_command*

```
-command command
[-task taskName | -script]
```

Data Types

```
command    string
command    taskName
```

Arguments

`-command command`

Specifies the Tcl command to execute and to add to the favorites and most-recently-used (MRU) palettes. The executed command is logged in the command log.

`-task taskName`

If this option is present, then the given task name is used when the command is added to the favorites and MRU palettes. If not present, then the current task name is used.

`-script`

This option is mutually exclusive with the -task option. If this option is present, the command is not run but is appended in the script editor.

Feedback

Description

This command executes the given Tcl command and records it in the command replay log. Use the *-script* option to append the command to the script editor only without executing it.

The result of a nested command is passed back as the result of *gui_eval_task_command*.

Examples

The following example evaluates the command "echo abc".

```
prompt> gui_eval_task_command -command {echo abc}
abc
```

The following example adds the command "echo abc" to the script editor.

```
prompt> gui_eval_task_command -command {echo abc} -script
abc
```

---

gui_execute_menu_item

Execute a menu item in the currently active window.

Syntax

```
status gui_execute_menu_item
{ -menu menu_item_id }
```

Data Types

```
menu_item_id      string
```

Arguments

`-menu menu_item_id`

menu_item_id specifies the hierarchical name of the menu item to execute. The "name" of a menu is a string which specifies the hierarchy of menu labels connected with '->'. For example, File->Quit specifies the Quit menu item in the File menu. These names are allowed to also specify the keyboard accelerator for the menu text which is specified with an embedded '&', but they are not required to do so.

Description

This command checks for the existence and enabled state of the given menu item in the currently active window. If it exists and is enabled, the menu item is executed.

The command returns and result value string from the executed menu item, if any.

Feedback

Examples

The following example activates the main window named Toplevel.2, then executes the *Hierarchy Browser* menu item in the *View* menu in Tolevel.2.

```
shell> gui_set_active_window -window Toplevel.2
shell> gui_execute_menu_item -menu "View->Hierarchy Browser"
```

See Also

- gui_create_menu
- gui_set_active_window

---

gui_exist_pref_category

Check the existence of a preference category.

Syntax

bool *gui_exist_pref_category* -category *Category*

```
Category        String
```

Arguments

`-category Category`

Category specifies the category.

Description

This command checks the existence of a preference category. Return "1" if the specified category exists, otherwise return "0".

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys

Feedback

- gui_exist_pref_key
- gui_update_pref_file

---

gui_exist_pref_key

Check the existence of a preference key.

Syntax

bool *gui_exist_pref_key* -key *Key*

[-category *Category*]

```
Key             String
Category        String
```

Arguments

`-key Key`

Key specifies the preference key of interest.

`-category Category`

Category specifies the category. Default category will be used if missing.

Description

This command checks the existence of a preference key given the key name and the preference category. Return "1" if the specified key exists, otherwise return "0".

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_update_pref_file

Feedback

gui_exist_window

Check for the existence of specified window or window type

Syntax

```
status gui_exist_window
{ -window window_id | -type window_type }
[ -parent window_id ]
```

Data Types

```
window_id       string
window_type     string
```

Arguments

`-window window_id`

Specifies the window id to test existence for.

This option is mutually exclusive with the *-type* option.

`-type window_type`

Specifies that the existence of a window of this window type is tested for.

If the *-parent* option is specified then the existence of a window of this window type is tested for in the specified toplevel window.

If the *-parent* option is *not* specified then the existence of a window of this window type is tested for in any toplevel window.

This option is mutually exclusive with the *-window* option.

`-parent window_id`

Specifies the toplevel window to check for existence in.

Note: This option is only relevant if the *-type* option is specified.

Description

This command checks the existence of the specified window, or window type across toplevel windows or within a specified toplevel window.

The command returns "1" or "0" for success or failure respectively.

Examples

The following example will create a toplevel window and a child layout view window, then check the existence of the command layout window.

Feedback

```
shell> set top [gui_create_window -type TopLevel]
shell> set x [gui_create_window -type Layout -parent $top]
shell> set layout_exist [gui_exist_window $x]
shell> if { $layout_exist == 1} {echo "layout exist" }
```

See Also

- gui_close_window
- gui_create_window
- gui_show_window
- gui_get_window_types
- gui_get_window_ids
- gui_set_active_window
- gui_get_current_window

---

gui_export_utable

Export the UserTable to a value file.

Syntax

string *gui_export_utable*

```
-name         Table_Name
[-file        File_Name [-tsv_mode] [-ssv_mode]]
[-filter      Filter]
[-tsv_mode]
[-ssv_mode]
[-add_col_type]
[-add_metadata]
[-overwrite]

Table_Name    String
File_Name     String
```

Arguments

`-name Table_Name`

The name of the user table to export to a CSV value file.

`-file File_Name`

The name of the file to write the table to, by default the name of the table is used with the '.csv' extension.

Feedback

`-filter Filter`

This argument allows you to filter the rows in the given table to only rows that match the filter. The filter expression is the same expression allowed in the GUI filter field of the UserTable GUI view.

`-tsv_mode`

This flag changes the default delimiter from the comma char to the tab char.

`-ssv_mode`

This flag changes the default delimiter from the comma char to the semicolon char.

`-add_col_type`

This flag will add column type information to be added as a postfix to the header column names. This allows type information to be available if imported again by this tool at a later time.

`-add_metadata`

Add metadata to the top of the exported table. See the command gui_set_utable_meta for more information.

`-overwrite`

This flag will force the destination file to be overwritten.

Description

```
This command exports the contents of the given UserTable into a CSV value
 file.
The first line contains all the column names and if the -add_col_type
 flag is
given, it also appends column data type information to the column names.
```

Examples

```
The following example just exports the given UserTable to a file with the
 same name
as the table plus a '.csv' extension.

shell> gui_export_utable -name my_table

The following example exports the UserTable 'my_table' to the filename
 'results' and
uses the Tab char as a delimiter. The resultant file will be called
 'results.tsv'.

shell> gui_export_utable -name my_table -file results -tsv_mode
```

Feedback

See Also

- gui_append_utable
- gui_change_selection_utable
- gui_close_utable
- gui_create_utable
- gui_import_utable
- gui_set_utable_meta
- gui_open_utable
- gui_show_utable
- gui_write_utable

---

gui_fill_utable

Fill generated data rows to an existing UserTable.

Syntax

string *gui_fill_utable*

```
-name              Name

-file_column       Name
-file_suffix       Suffix
-file_dir          Directory
-label_column      Name
-label_value       Value
-date_column       Name
-tag_column        Name
-title_column      Name

Name               String
File_Name          String
Suffix             File Suffix String
Directory          Directory Path String
Value              Value String
```

Arguments

`-name Name`

The name of an existing UserTable that the new data rows should be added to.

Feedback

`-file_column` *Directory*

The column name that is filled with the filenames found in the directory.

`-file_dir` *Directory*

The directory path to search for files matching the file suffix option.

`-label_column` *Name*

The column name that is filled with a label for each entry added.

`-label_value` *Value*

The label value used to fill the -label_column argument as each entry is added.

`-date_column` *Name*

The column name used to fill in the date of the file that was added.

`-tag_column` *Name*

The column name used to fill in the metadata tag of the file if it exists.

`-title_column` *Name*

The column name used to fill in the metadata title of the file if it exists.

Description

Using the various options one can generate rows of data that include columns for file names, file dates, labels and if available in the file the tag and title meta data for all files found according to the specified file suffix and directory. For more information on Meta Data please see the command *gui_set_utable_meta*.

Examples

The following example creates a table of data where each row contains information about a CSV file that was found in the given directory. The filename is actual a link that when clicked will load that CSV file and make it a child of this table. Thus this basically creates a table of links to other tables represented by CSV files.

```
shell> gui_create_utable -name csv_toc \\
         -columns {status:boolean comment report file date tag title} \\

shell> gui_fill_utable -name csv_toc \\
         -file_column file \\
         -file_suffix ".csv" \\
         -file_dir "./csv_files/" \\
         -label_column report \\
```

Feedback

```
        -label_value value_files \\
        -date_column date \\
        -tag_column tag \\
        -title_column title
```

See Also

- gui_append_utable
- gui_close_utable
- gui_change_selection_utable
- gui_create_utable
- gui_export_utable
- gui_get_utable
- gui_import_utable
- gui_set_utable_meta
- gui_open_utable
- gui_show_utable
- gui_write_utable

---

gui_foreach_utable_row

Iterate/visit the given table row column values for the given filter and call a TCL proc.

Syntax

string *gui_foreach_utable_row*

```
-name             Table_Name
-action           Action
-columns          Column_List
[-filter          Filter_Expr]
[-data            Data]

Table_Name       String
Action           TCL Procedure Name
Column_List      StringList
Filter_Expr      String
Data             String
```

Feedback

Arguments

`-name` *`Table_Name`*

The table name on which to visit all the rows filtered by the filter.

`-action` *`Action`*

The user TCL procedure that will be called on every row that matches the filter.

`-columns` *`Column_List`*

Provides a list of one or more column names whose data will be passed to the action procedure.

`-filter` *`Filter_Expr`*

To only visit a certain set of rows in the table you must ether first filter the table in the view or use this filter expression to only target the rows of interest.

`-data String`

Provides an optional data string that is passed to the user TCl proc.

Description

```
This command allows you iterate/visit each row in a table filtered
by the filter value and call a user given TCL procedure.
The procedure must accept two fixed arguments, the 'table_name'
and a 'user_data' argument which may be filled with the '-data' option.
This is then followed by an argument for every column name given which
will have the current row value for that column.
If no filter option is given the all rows will be visited unless the
table is currently viewed, then all rows will be visited that match
the current view filter setting.
```

Examples

```
The following example calls a print TCL procedure for every row that
matches the filter.

shell>
  proc printVal { table data pin direction } {
    puts "table: $table, data: $data, pin: $pin, direction: $direction"
  }

  gui_foreach_utable_row -name $table \\
    -columns {{full_name} {direction}} \\
    -action printVal -filter {slack > 0.0}

The following example uses the gui_set_utable_values command to update
the table values. This is not as efficient as calling the
 gui_set_utable_values
command for all filtered values but it gives you more control for each
row update action.
```

Feedback

```
shell>
  proc setVal { table data pin direction } {
    # Do some arbritary functions with the received data...
    # Update the table row itself:
    gui_set_utable_values -name $table -column direction \\
      -value $data -filter "{full_name} == $pin"
  }
  gui_foreach_utable_row -name $table \\
      -data out \\
      -columns {{full_name} {direction}} \\
      -action setVal -filter {slack > 0.0}
```

See Also

- gui_get_utable
- gui_set_utable
- gui_set_utable_values

---

gui_get_annotations

Return a collection of layout annotations

Syntax

string *gui_get_annotations*

[-window *window_name*] [-group *group_id*] [-within *region*] [-intersect] [-at *location*] [-filter *filter*] [-nocase] [-regexp]

```
string window_name
string group_name
string filter
rectangle region
location point
```

Arguments

`-window window_name`

Specifies the instance name of the layout window. If not specified implies all windows.

`-group group_name`

Specifies the annotation group name. If not specified implies all groups.

`-within region`

Search for annotations contained within the specified rectangle.

Feedback

`-intersect`

Use this option with the -within option. When -intersect is specified, all annotations that touch the search rectangle are also returned.

`-at` *`location`*

Search for annotations that are close to the specified point.

`-filter` *`filter`*

Only return annotations that satisfy the filter expression.

`-nocase`

Ignore case in the filter expression. Option requires -filter.

`-regexp`

Use regular expressions in the filter expression. Option requires -filter.

Description

Returns a collection of gui_annotation objects that satisfy the requirements. You can use get_attribute to query properties of the annotations. Also some attributes may be modified with set_attribute.

Examples

The following example retrieves all the annotations in the Layout.1 window.

```
shell> gui_get_annotations -window Layout.1
```

This example shows how to use the client_data attribute to find annotations with that setting.

```
shell> gui_get_annotations -filter client_data==MyData
```

This example shows how to find annotations that are within a specified rectangle

```
shell> gui_get_annotations -within {{10.5 2.0} {23.5 14.6}}
```

To also find the annotations that touch the search rectangle:

```
shell> gui_get_annotations -within {{10.5 2.0} {23.5 14.6}} -intersect
```

gui_get_bucket_option

Returns the value of a bucket option of a given visual or map mode.

Syntax

```
string gui_get_bucket_option
-map map_name
```

Feedback

```
-bucket bucket_name
-option option_name
[-default]
```

Data Types

```
map_name          string
bucket_name       string
option_name       string
```

Arguments

`-map map_name`

Specifies the name of the visual mode or map mode.

`-bucket bucket_name`

Specifies the name of the bucket.

`-option option_name`

Specifies the name of the option to be set.

`[-default]`

Specifies that command has to return default option value.

Description

This command returns an option value that are available for the bucket in the specified visual mode or map mode. You must specify the visual mode or map mode name, the bucket name, and the option name. The command can return either current or default value.

Examples

The following example returns the value of 'visible' option of 'light blue' bucket of 'Highlight' visual mode:

```
prompt> gui_get_bucket_option -map {Highlight} -bucket {light blue}
 -option {visible}
```

See Also

- gui_set_bucket_option
- gui_get_bucket_option_list
- gui_get_map_list
- gui_get_map_option

Feedback

- gui_get_map_option_list
- gui_set_map_option

---

gui_get_bucket_option_list

Lists the available options for buckets in the specified visual mode or map mode.

Syntax

```
string gui_get_bucket_option_list
-map map_name
```

Data Types

map_name	string

Arguments

`-map` *map_name*

Specifies the name of the visual mode or map mode.

Description

This command displays a list of all options that are available for the buckets in the specified visual mode or map mode.

Examples

The following example displays a list of the available options for buckets in the global route congestion map:

```
prompt> gui_get_bucket_option_list -map AREAPARTITION
```

See Also

- gui_get_bucket_option
- gui_get_map_list
- gui_get_map_option
- gui_get_map_option_list
- gui_set_bucket_option
- gui_set_map_option

Feedback

gui_get_category

Returns the contents of the specified category as a collection, including the contents of any offspring categories.

Syntax

collection *gui_get_category*

```
-category category_hierarchical_name
[-tree tree_id]
```

Data Types

category_hierarchical_name	list
tree_id	string

Arguments

`-category` *category_hierarchical_name*

Specifies the hierarchical name of the category whose contents are returned as a collection. The *category_hierarchical_name* value is a unique hierarchical name of a category in Tool Command Language (Tcl) list format.

`-tree` *tree_id*

Specifies the string ID of the category tree in which the get-category operation occurs. The current category tree is used if the *-tree* option is not specified.

Description

This command returns the contents of the specified category as a collection, including the contents of any direct or indirect offspring categories.

You should use this command only if at least one category tree is available. The command acts on the specified category tree.

Examples

The following example gets the contents of the ALL root category as a collection, including the contents of all direct or indirect offspring categories of the ALL root category, and captures the collection returned by the *gui_get_category* command in a Tcl variable:

```
prompt> set category_collection [gui_get_category -category {ALL}]
```

The following example gets the contents of the {Launch Clock: CLK1} category. This category is a child of the {Pathgroup: CLK1} category, which in turn is a child of the ALL root category.

```
prompt> set clct [gui_get_category \\
        -category {ALL {Pathgroup: CLK1} {Launch Clock: CLK1}}]
```

Feedback

See Also

- gui_create_category_rule
- gui_list_category_rules
- gui_remove_category_rules

---

gui_get_cell_block_marks

Gets a list of the block mark string values on one or more cells.

Syntax

list *gui_get_cell_block_marks*

```
        cells
```

Data Types

```
cells       list
```

Arguments

cells

Lists one or more cell name patterns or cell collections.

Description

This command gets the block mark string values from one or more hierarchical or leaf-level cell instances.

Examples

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the *get_cells* command, and then gets the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU]  ALU
prompt> gui_get_cell_block_marks [get_cells I_TOP/I_ALU]
ALU
```

Feedback

The following example shows how using the *get_attribute* command to query the *block_mark* attribute is an alternative to using the *gui_get_cell_block_marks* command:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_get_cell_block_marks I_TOP/I_ALU
ALU
prompt> get_attribute -class cell I_TOP/I_ALU block_mark
ALU
```

See Also

- gui_set_cell_block_marks
- gui_remove_cell_block_marks

---

gui_get_charts_data

Get charts data

Syntax

```
value gui_get_charts_data
{ -global | -model model_id | -view view_name |
-plot plot_name | -annotation annotation_name }
[ -column column_name ]
[ { -header | -row int } ]
-name string
[ -data string ]
```

Data Types

```
model_id          int
view_name         string
plot_name         string
annotation_name   string
column_name       int
value               string
```

Arguments

`-global`

Get global data.

`-model model_id`

Model to get data from.

`-view view_name`

View to get data from.

`-plot` *`plot_name`*

Plot to get data from.

`-annotation` *`annotation_name`*

Annotation to get data from.

`-column` *`column_name`*

Column to get model data from.

Only used by the *model* option.

`-header`

get model data from header.

Only used by the *model* option.

`-row` *`int`*

Row to get model data from.

Only used by the *model* option.

`-name` *`string`*

Name of data to return.

The supported names depend on which object the data is extracted from:

For model the following names are supported:

value Get value of model horizontal header (using the *header* option) value or row value (using the *row* option). The column is specified using the *column* option.

num_rows Get number of rows in model.

num_columns Get number of columns in model.

header Get model header names. If column is specified using the *column* option then only the header for that column is returned.

row Get model row values for the row specified using *row* option.

column Get model column values for the column specified using *column* option.

duplicates Get rows with duplicate values (same value in all columns).

column_index Get model column index from name specified in *data* option.

name Get the model name.

title Get the model title.

Feedback

The following data names are queried on the specifed column (single value) or, if the column is not specified, then on all columns (list of values):

type Get model column type.

min or minimum Get model minimum column value.

max or maximum Get model maximum column value.

mean or avg or average Get model average column value.

monotonic Get if model column monotonic (increasing or descreasing)

increasing Get if model column increasing

num_unique Get number of unique model values in column.

unique_values Get unique model values in column.

unique_counts Get counts of each unique model value in column.

num_null Get number of null values in column.

median Get model median column value.

lower_median Get model lower median column value.

upper_median Get model upper median column value.

stddev or std_dev Get model standard deviation value.

outliers Get model outlier values.

For view the following names are supported:

plots return a list of all plots in the view.

annotations return a list of all annotations in the view.

selected_objects return a list of all selected objects in the view.

For plot the following names are supported:

model Model index.

view Parent view name.

annotations Get plot's annotation object ids.

objects Get plot's object ids.

selected_objects Get plot's selected object ids.

errors Get plot's error messages.

Feedback

For annotation the following names are supported:

view Parent view name.

plot Parent plot name.

The following global data the following names are supported:

models Get model indices

views Get view names.

plot_types Get plot type names.

plots Get all plots names.

column_types Get all column type names.

column_type.names Get names of parameters for specified column type (in *data* option).

annotation_types Get all annotation type names.

symbol_names Get all symbol names.

procs Get all user defined procedure names.

You can also use the special name "?" which will return a list of the supported names for the specified object.

`-data` *`string`*

Extra data needed to query some data values.

Returns

value

Returned value.

Description

Get data from model, view, plot type, plot objects or from global object.

If model is specified then the *column* and *header* or *row* should be specified.

Examples

The following example gets all the view names.

```
shell> set views [gui_get_charts_data -global -name views]
```

The following example gets all the plots of the first view.

Feedback

```
shell> set plots [gui_get_charts_data -view [lindex $views 0] -name
 plots]
```

The following example gets all the annotations of the first plot.

```
shell> gui_get_charts_data -plot [lindex $plots 0] -name annotations
```

The following example gets all the model names.

```
shell> set models [gui_get_charts_data -global -name models]
```

The following example gets the value at row 2, column 1 in the first mode.

```
shell> gui_get_charts_data -model [lindex $models 0] -name value -row 2
 -column 1
```

See Also

- gui_set_charts_data

---

gui_get_charts_property

Get charts property

Syntax

```
value gui_get_charts_property
{ -view view_name | -plot plot_name |
-annotation annotation_id }
-name string
```

Data Types

```
view_name       string
plot_name       string
annotation_id   string
value             string
```

Arguments

`-view view_name`

View to get property from.

`-plot plot_name`

Plot to get property from.

`-annotation annotation_id`

Plot or view annotation to get property from.

Feedback

`-name string`

Name of property to return.

The supported names depend on which object the property is extracted from but can be listed by using the special property name "?".

Returns

value

Returned value.

Description

Get property from view, plot, view annotation, plot annotation or plot objects.

Examples

The following example gets the position of an annotation in a plot.

```
shell> gui_get_charts_property -plot $plot -annotation $annotation -name
 position
```

See Also

- gui_set_charts_property

---

gui_get_color_value

Get the RGB color value for given color index or name, else return all color information for the palette.

Syntax

string *gui_get_color_value [index]*

```
[-palette string]
[-color_name string]
```

Arguments

index

Use given color index number. An error message is posted if the given index is greater than the maximum index allowed for the current or given Palette. This argument is exclusive with option -color_name.

-palette

User given palette name, the default is 'highcontrast'. Valid values are: highcontrast, highlight, default_tech, qt, style. An error message is printed if an invalid palette name is given.

-color_name

Use given color name. Must be valid for the current/given palette. If an invalid color name is given an error message is posted. This option is exclusive with the 'index' argument.

Description

Get the RGB color value string for the given color name or index value.

If no argument is given the command will return a TCL list of lists. One list for every color in the current/given palette. The list for each color contains the index number, the RGB color value and, if available, a color name.

Examples

```
shell> gui_get_color_value 10
#005000

shell> gui_get_color_value -color_name magenta
#ff00ff

shell> gui_get_color_value
{0 #000000 }
{1 #464646 }
{2 #00ff64 }
{3 #78bec8 }
{4 #ff00ff magenta}
{5 #00af96 }
{6 #0000be }
{7 #00ffff cyan}
{8 #b1f476 light_green}
{...}
```

---

gui_get_current_task

Get the name of the current task.

Syntax

string *gui_get_current_task*

Description

The gui_get_current_task command queries the name of the current task.

Feedback

See Also

- gui_create_task
- gui_get_current_task_item
- gui_get_current_task_page
- gui_set_current_task
- gui_get_task_list

---

gui_get_current_task_item

Get the name of the current task item.

Syntax

string *gui_get_current_task_item*

Description

The gui_get_current_task_item command queries the name of the current task item.

See Also

- gui_get_current_task
- gui_get_current_task_page
- gui_set_current_task
- gui_get_task_list

---

gui_get_current_task_page

Get the name of the current task page.

Syntax

string *gui_get_current_task page*

Description

The gui_get_current_task_page command queries the name of the current task page.

See Also

- gui_create_task
- gui_get_current_task

Feedback

- gui_get_current_task_item
- gui_set_current_task
- gui_get_task_list

---

gui_get_current_window

Retrieves the window ID of the active top-level or view window.

Syntax

```
string gui_get_current_window
[ -toplevel | -view | -parent parent_window_id ]
[ -hier_name ]
[ -types window_type_list ]
[ -mru ]
```

Data Types

```
parent_window_id    string
window_type_list    list
```

Arguments

`-toplevel`

Retrieves the active top-level window unless you also specify the *-types* option.

If you also specify the *-types* option, the *-toplevel* option retrieves the most recently active top-level window that has one of the specified window types.

The *-toplevel*, *-view*, and *-parent* options are all mutually exclusive.

`-view`

Retrieves the active view window in the active top-level window unless you also specify the *-types* option, the *-mru* option, or both.

if you also specify the *-types* option but not the *-mru* option, the *-view* option retrieves the most recently active view window with one of the specified window types in the active top-level window.

if you also specify the *-mru* option but not the *-types* option, the *-view* option retrieves the most recently active view window in any of the open top-level windows.

If you also specify both the *-types* option and the *-mru* option, the *-view* option retrieves the most recently active view window that has one of the specified window types and is in an open top-level window.

The *-view*, *-toplevel*, and *-parent* options are all mutually exclusive.

`-parent` *`parent_window_id`*

Retrieves the active view window in the specified top-level window unless you also specify the *-types* option.

If you also specify the *-types* option, the *-parent* option retrieves the most recently active view window that has any of the specified window types and is in the specified top-level window.

The *-parent*, *-toplevel*, and *-view* options are all mutually exclusive.

`-hier_name`

Returns the window ID in Qtcl format, which is similar to a full path name for the window, for use with the qtcl_* commands.

Note that a value returned in this format is not usable by gui_* commands.

`-types` *`window_type_list`*

Forces the tool to retrieve a window with one of the specified window types.

Use this option to restrict the search to certain types of windows. If you specify multiple types, they should be consistent (all toplevel window types or all view window types), but nonconsistency is not considered an error. The first type in the list determines whether the tool retrieves a top-level window or a view window when the *-toplevel* and *-view* options are not specified.

When you use the *-types* option, you should not use the *-toplevel* option or the *-view* option because the tool can determine from the list whether the window should be a top-level window or a view window. If you specify one of these options, any types that do not correspond to the option are silently ignored. For example, if you specify *-view*, then all top-level types are ignored.

`-mru`

Forces the tool to retrieve the most recently active view window from among all the open top-level and view windows. This option is effective only when you specify the *-view* option or when all the window types you specify with the *-types* option are view window types.

Description

This command retrieves the window ID of the active top-level or view window based on the options that you specify, or returns an empty string if no match is found.

A view window is a child window of a top-level window and is created with the *gui_create_window* command.

If you do not specify the *-toplevel*, *-view*, and *-parent* options, the tool uses the *-types* option to determine the window type. If you also do not specify the *-types* option, the tool uses the *-toplevel* option by default.

Feedback

Examples

The following example retrieves the active top-level window:

```
prompt> gui_get_current_window -toplevel
```

The following example retrieves the active view window:

```
prompt> gui_get_current_window -view
```

The following example retrieves the most recently active layout view:

```
prompt> gui_get_current_window -types "Layout" -mru
```

The following example return $cons:

```
prompt> set top1 [gui_create_window -type LayoutWindow]
prompt> set top2 [gui_create_window -type LayoutWindow]
prompt> set cons [gui_create_window -type Layout -parent $top]
prompt> gui_get_current_window -parent $top
```

See Also

- gui_close_window
- gui_exist_window
- gui_show_window
- gui_get_window_types
- gui_get_window_ids
- gui_set_active_window
- gui_create_window

---

gui_get_eco_context

Gets the context for manual ECO in the GUI for a given type if specified or for all the types.

Syntax

```
status gui_get_eco_context
[-type ype of context]
```

Data Types

type	string

Feedback

Arguments

type

Gets the context for the given type. Supported types are 'insert_buffer' or 'size_cell'. If no type is specified, it will get you both the types.

Description

The *gui_get_eco_context* command sets the context of manual ECO in the GUI.

The *gui_reset_eco_context* command can be used to reset the context and *gui_set_eco_context* can be used to set the context in the GUI for manual ECO

Examples

The following example shows how to get a given context.

```
prompt> gui_get_eco_context -type insert_buffer
1
prompt> gui_get_eco_context
1
```

See Also

- gui_set_eco_context
- gui_reset_eco_context

---

gui_get_error_browser_option

Get Error Browser Dialog option values

Syntax

string *gui_get_error_browser_option*

-grouping | -show_mode | -view_mode | -zoom_factor | -hide_fixed | -hide_ignored | -advance_to_next_unfixed_error | -dim | -show_open_locator_nodes | -show_tooltip | -show_command_buttons | -highlight_objects | -auto_set_object_visible | -auto_set_object_visible_type

Arguments

```
-grouping | -show_mode | -view_mode | -zoom_factor | -hide_fixed
| -hide_ignored | -advance_to_next_unfixed_error | -dim |
-show_open_locator_nodes | -show_tooltip -show_command_buttons
```

Feedback

```
| -highlight_objects | -auto_set_object_visible |
-auto_set_object_visible_type
```

Mutually exclusive required option to specify the Error Browser option value to query.

Description

This command gets the value of the requested option setting for the Error Browser. One option value can be queried at a time.

The *-grouping* option returns *type* or *layer* which indicates whether errors are grouped by types or layers.

The *-show_mode* option returns *all*, *selected*, or *none* which indicates whether all errors, selected errors, are no errors are shown in layout views.

The *-view_mode* option returns *zoom*, *pan*, of *off* which indicates whether the layout view zooms to, pans to, or does not change to the currently selected errors in the Error Browser.

The *-zoom_factor* option returns a float value between *0.1* and *10.0* and indicates the zoom factor used when the view_mode is zoom.

The *-hide_fixed* option returns *true* to mean "hidden" or *false* to mean "not hidden" and indicates whether fixed errors are hidden from display in the Error Browser and layout views.

The *-hide_ignored* option returns *true* to mean "hidden" or *false* to mean "not hidden" and indicates whether ignored errors are hidden from display in the Error Browser and layout views.

The *-advance_to_next_unfixed* option returns *true* to mean advance to next unfixed error when an error is fixed or *false* to mean advanced to next fixed/unfixed error.

The *-dim* option returns *true* to mean "dimmed" or *false* to mean "not dimmed" and indicates whether layout views are dimmed when errors are displayed.

The *-show_open_locator_nodes* returns *true* to mean "highlighted" or *false* to mean "not highlighted" and indicates whether net nodes are highlighted along with the open locator flylines for selected open locator errors.

The *-show_tooltip* returns *true* to mean show tooltips or *false* to mean don't show tooltips in the error browser list and details panes.

The *-show_command_buttons* returns *true* to mean show the commannd buttons or *false* to mean don't show the command buttons in the error browser.

The *-highlight_objects* returns *true* to mean "highlighted" or *false* to mean "not highlighted" and indicates whether associated design objects are highlighted in the layout view for selected errors.

Feedback

The *-auto_set_object_visible* returns whether auto set layout/object visibility is turned on or not.

The *-auto_set_object_visible_type* returns whether auto set layout/object visibility turns on layers/objects incrementally (the default) or exclusively.

Examples

The following example gets the grouping option

gui_get_error_browser_option -grouping

The following example gets whether layout views are dimmed when errors are displayed

gui_get_error_browser_option -dim

See Also

- gui_error_browser
- gui_set_error_browser_option

---

gui_get_error_data

Creates a collection of physical DRC error data that are currently shown in the error browser.

Syntax

```
collection gui_get_error_data
```

Description

The *gui_get_error_data* command creates a collection of physical DRC error data that are currently shown in the error browser.

You can use the *get_drc_error_data* command at the command prompt, or you can nest it as an argument to another command (for example, *query_objects*). In addition, you can assign the *gui_get_error_data* result to a variable.

When issued from the command prompt, *gui_get_error_data* behaves as though *query_objects* had been called to display the objects in the collection.

Examples

The following example queries the physical DRC error data that are currently shown in the error browser. Although the output looks like a list, it is not. The output is just a display.

```
prompt> gui_get_drc_error_data
{"my_design_dppinassgn.err"}
```

Feedback

See Also

- gui_open_error_data
- gui_close_error_data

---

gui_get_errors

Creates a collection of violation objects or errors in the Error Browser.

Syntax

collection *gui_get_errors*

[-visible] [-current] [-selected] [-fixed 0 | 1 | true | false] [-ignored 0 | 1 | true | false] [-id *ErrorObjectIDList*] [-data_name *ErrorDataName*]

```
ErrorObjectIDList   list
ErrorDataName       string
```

Arguments

`-visible`

Return errors that are currently visible in the Error Browser. This option will be exclusive to *-data_name* and *-id*.

`-current`

Return errors in the current error set, visible and hidden in the Error Browser. This option will be exclusive to *-data_name* and *-id*.

`-selected`

Return selected errors. This option will be exclusive to *-data_name* and *-id*.

`-fixed`

If this option is present, will only get errors that match the specified fixed state.

`-ignored`

If this option is present, will only get errors that match the specified ignored state.

`-id`

Return errors specified by error object ids. It will only work with *-data_name* and exclusive to *-visible*, *-current* and *-selected*.

`-data_name` *`ErrorDataName`*

ErrorDataName specifies the error data to get error objects. It will only work with *-id* and exclusive to *-visible*, *-current* and *-selected*.

Feedback

The error data name may be omitted. If omitted, selects the design, the parent of all open error data.

Description

This command creates a collection of violation objects or errors in the Error Browser.

If *-visible* option specified, return all visible errors in the Error Browser. If *-current* option specified, return all errors in the current error set. If *-selected* option specified, return all selected errors. *-visible*, *-current* and *-selected* will be exclusive to *-id* and *-data_name*. If specified together, an error message will be issued.

If *-fixed* option specified, if true then return all fixed errors, else return all non-fixed errors. If *-ignored* option specified, if true then return all ignored errors, else return all non-ignored errors. If *-fixed* and *-ignored* are specified together, return a combination of each result.

If *-id* option specified, return all errors based on the id list from specified error data. If *-data_name* option specified, return errors from the specified data. If the *-data_name* option is omitted, selects the design, the parent of all open error data. *-id* and *-data_name* have to be specified together. They will be exclusive to *-visible*, *-current* and *-selected*.

Examples

The following example gets errors from current error set and with fixed status 1.

```
gui_get_errors -current -fixed 1
```

The following example gets errors from zroute.err with id 1 and 2.

```
gui_get_errors -data_name zroute.err -id {1, 2}
```

See Also

- gui_error_browser
- gui_set_selected_errors
- gui_clear_selected_errors
- gui_zoom_to_selected_errors

---

gui_get_hierview_data

Get various state about the Hierarchy View and its sub-views. [Default return last used Hierarchy View name]

Syntax

string *gui_get_hierview_data*

Feedback

```
[-view              View_Name]
[-tree_type]
[-tree_attr_group]
[-tree_filter]
[-map_area]
[-map_color]
[-childlist_name]
[-childlist_attr_group]
[-childlist_filter]
[-sub_view]
[-columns]
[-values            Column_Name]
[-elide]
[-attr]

View_Name      String
Column_Name    String
```

Arguments

`-view View_Name`

Use the given HierView window name (ex: Hier.1). Default is to get the most recent used HierView window name.

`-tree_type`

Get the current Hier Tree type (logical | rtl). Note: RTL is only available in RTLA.

`-tree_attr_group`

Get and return the current Hier Tree attribute group name.

`-tree_filter`

Get and return the current Hier Tree filter expression.

`-map_area`

Get and return the Hier Map Area attribute group name.

`-map_color`

Get and return the Hier Map Color attribute group name.

`-childlist_name`

Get and return the current ChildList sub-view name.

`-childlist_attr_group`

Get and return the current ChildList attribute group name.

`-childlist_filter`

Get and return the current ChildList filter expression.

Feedback

`-sub_view`

Get and return the current active sub-view type. The sub-view types are 'hier_tree', 'hier_map' and 'child_list'.

`-columns`

Get and return a list of all column name(s) of all selected table cells.

`-values` *`Column_Name`*

Get and return a list of all selected table cell value(s) in the given column name. *Note*: Any of the column value(s) in the row of the selected cell(s) can be retrieved.

`-elide`

Get and return the current text elide for the view.

`-attr`

Get the attribute name for the given column short name (display label).

Description

```
This command allows you to get various current Hierarchy View related
 view settings
and selected table cell data.
```

See Also

- gui_set_hierview_data

---

gui_get_highlight

Get a collection of highlighted objects.

Syntax

string *gui_get_highlight*

```
[-color color_id | -all_colors]
[-return_select_bus | -more_than ]

string          color_id
collection      clct
```

Arguments

`-color` *`color_id`*

Specifies the color of objects to be returned. If neither -color nor -all_colors is used, the current color is assumed. The allowed colors are yellow, orange,

Feedback

red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

`-all_colors`

Specifies that all highlighted objects are to be returned.

`-return_select_bus`

Create a new selection bus in which to store the result instead of returning a collection of objects.

`-more_than`

Return 1 if more than the specified number of objects would be returned, otherwise 0.

Description

The gui_get_highlight command returns a collection of objects highlighted with the specified colors.

Examples

Get a collection of green highlighted objects.

shell> gui_get_highlight -color green

Get a collection of all highlighted objects.

shell> gui_get_highlight -all_colors

Query if any objects are highlighted

shell> gui_get_highlight -all_colors -more_than 0

See Also

- gui_change_highlight
- gui_get_highlight_options
- gui_set_highlight_options

---

gui_get_highlight_options

Query the options that control highlighting.

Syntax

string *gui_get_highlight_options*

`[-current_color | -all_colors | -auto_cycle_color]`

Feedback

Arguments

`-current_color`

This option returns a string which is the current highlight color, or the default color for highlighting operations.

`-all_colors`

This option returns Tcl list of all of the colors which are allowed to be used for highlighting.

`-auto_cycle_color`

This option returns the current value of the auto cycle setting ("true" or "false").

Description

The gui_get_highlight_options command queries the settings which control highlighting. Exactly one option must be provided.

Examples

Get the current highlight color.

shell> gui_get_highlight_options -current_color

Get all highlight colors.

shell> gui_get_highlight_options -all_colors

See Also

- gui_change_highlight
- gui_get_highlight
- gui_set_highlight_options

---

gui_get_map

Retrieves attributes for a specified map mode.

Syntax

string *gui_get_map*

```
-name identifier
[-buckets]
[-title]
[-help_topic]
[-infotip]
[-discrete]
```

```
[-icon_file]
[-update_cmd]
[-float]
[-top_exaggeration]
[-mid_exaggeration]
[-bot_exaggeration]
```

Data Types

identifier string

Arguments

-name *identifier*

Specifies the name of the map mode to be queried.

-buckets

Returns a list of bucket names. The bucket names are retrieved in rendering order, starting from the first to the last bucket rendered.

-title

Returns the title string.

-help_topic

Returns the help topic string.

-infotip

Returns the InfoTip string.

-discrete

Returns a true or false value that indicates whether the buckets of this map mode are discrete or continuous. Only discrete buckets can be reordered. This attribute is set when this map mode was created, and cannot be changed.

-icon_file

Returns the gui menu icon file.

-update_cmd

Returns the Tcl command that is executed when the map mode is accessed. A typical use for this option is to update the state of the map mode if its contents are likely to change under certain circumstances.

-float

Returns a true or false value indicating whether the map mode buckets have a floating point range.

Feedback

`-top_exaggeration`

Returns the top bucket exaggeration.

`-mid_exaggeration`

Returns the middle bucket exaggeration.

`-bot_exaggeration`

Returns the bottom bucket exaggeration.

Description

The *gui_get_map* command queries the option values of an existing map mode.

Examples

To retrieve names of the buckets in the map mode named "densityMap", use the following command:

```
prompt>gui_get_map -name densityMap -buckets
```

See Also

- gui_get_mapbucket

---

gui_get_map_list

Lists the current visual and map modes.

Syntax

```
string gui_get_map_list
```

Arguments

The *gui_get_map_list* command has no arguments.

Description

The *gui_get_map_list* command returns a list of existing visual and map modes.

Examples

The following example gets the names of all available visual and map modes:

```
prompt> gui_get_map_list
```

Feedback

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_bucket_option
- gui_get_bucket_option_list
- gui_get_map_option
- gui_get_map_option_list
- gui_get_vm
- gui_get_vmbucket
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_bucket_option
- gui_set_map_option
- gui_set_vm
- gui_set_vmbucket
- gui_show_map
- gui_update_vm

---

gui_get_map_option

Gets the value for an option in the specified visual mode or map mode.

Syntax

```
string gui_get_map_option
-map map_name
-option option_name
[-default]
```

Data Types

```
map_name        string
option_name     string
```

Feedback

Arguments

`-map map_name`

Specifies the name of the visual mode or map mode.

`-option option_name`

Specifies the name of the option to be set.

`[-default]`

Gets the option default value.

Description

This command gets the value for an option in a visual mode or map mode. You must specify the visual mode or map mode name and the option name. You can get set option specific value or its default value.

Examples

The following example gets the number of bins in the global route congestion map:

```
prompt> gui_get_map_option -map AREAPARTITION \\
        -option num_bins -value 9
```

See Also

- gui_get_bucket_option
- gui_get_bucket_option_list
- gui_get_map_list
- gui_set_map_option
- gui_get_map_option_list
- gui_set_bucket_option

---

gui_get_map_option_list

Lists the available options for the specified visual mode or map mode.

Syntax

```
string gui_get_map_option_list
-map map_name
```

Data Types

```
map_name        string
```

Feedback

Arguments

`-map map_name`

Specifies the name of the visual mode or map mode.

Description

This command displays a list of all options that are available for the specified visual mode or map mode.

Examples

The following example displays a list of the available options for the global route congestion map:

```
prompt> gui_get_map_option_list -map AREAPARTITION
```

See Also

- gui_get_bucket_option
- gui_get_bucket_option_list
- gui_get_map_list
- gui_get_map_option
- gui_set_bucket_option
- gui_set_map_option

---

gui_get_mapbucket

Gets attributes for Map Mode Bucket

Syntax

string *gui_get_mapbucket*

```
-map mode_identifier
-name bucket_identifier
[-infotip]
[-color]
[-pattern]
[-exaggeration]
[-number]
[-maxval]
[-minval]
[-visible]
[-special]
```

Feedback

```
[-title]
[-objcount]
```

Data Types

```
mode_identifier   string
bucket_identifier string
```

Arguments

`-map` *mode_identifier*

Specifies the map mode to which the bucket is a member. The map mode must already exist.

`-name` *bucket_identifier*

Specifies the name of the map mode bucket to be queried. To see the list of buckets associated with a specific map mode use the gui_get_map command with the -buckets option.

`-infotip`

Return infotip string.

`-color`

Return bucket rendering color.

`-pattern`

Return bucket rendering fill pattern.

`-exaggeration`

Return bucket min pixel exaggeration value.

`-number`

Return bucket display number.

`-maxval`

Return bucket maximum value.

`-minval`

Return bucket minimum value.

`-visible`

Return visibility.

`-special`

Return whether bucket is special.

Feedback

`-title`

Return bucket title.

`-objcount`

Return the number of objects in the bucket.

Description

The gui_get_mapbucket command queries an instance of a map mode bucket for the specified option values. The values are returned in a list of option-value pairs.

Examples

Query the current color for the bucket named "Bucket10" in the map mode "densityMap":

```
prompt> gui_get_mapbucket -map densityMap -name Bucket10 -color
```

Query the current color and pattern for the bucket named "Bucket10" in the map mode "densityMap":

```
prompt> gui_get_mapbucket -map densityMap -name Bucket10 -color -pattern
```

See Also

- gui_get_map

---

gui_get_menu_roots

Return a sorted list of all menu roots used by the application.

Description

This command returns a list of all the menu roots used by the application. The menu roots include menu roots used by toplevel menu bars and menu roots used by context menus. A menu root is used to group menu items that share the same menu root together. All items under each toplevel window's menu bar shares the same menu root; similarly, all items in a context menu (menu brought up with right mouse click in a window) shares the same context menu root.

See Also

- gui_create_menu

---

gui_get_mouse_tool_option

Get the value of a mouse tool option

Feedback

Syntax

string *gui_get_mouse_tool_option* -tool *string*

-option *string*

```
string string
string string
```

Arguments

`-tool string`

Tool to get option for.

`-option string`

Option to get.

Description

This command returns value of a mouse tool option .

Examples

```
> gui_get_mouse_tool_option -tool SELECT -option InputMode
Smart
```

See Also

- gui_mouse_tool
- gui_set_mouse_tool_option

---

gui_get_performance_log_option

Gets the current option value controlling performance log behavior.

Syntax

`string` **`gui_get_performance_log_option`**

Description

This command is used to query the current setting for the performance log which is set by gui_set_performance_log_option.

Examples

```
The following example sets, then gets the current setting.

prompt> gui_set_performance_log_option -commands {gui_zoom
 win_select_objects}
```

Feedback

```
prompt> gui_get_performance_log_option
commands: gui_zoom win_select_objects
```

See Also

- gui_log_performance
- gui_set_performance_log_option

---

gui_get_pref_categories

Return a list of the current preference categories.

Syntax

string *gui_get_pref_categories*

Description

This command returns a tcl list of the current preference categories.

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_get_pref_keys

Return a list of preference keys under the specified category.

Syntax

string *gui_get_pref_keys* -category *Category*

Feedback

```
Category        String
```

Arguments

```
-category Category
```

Category specifies the category to use.

Description

Return a list of preference keys under the specified category.

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_get_pref_value

Get the value of a preference key.

Syntax

string *gui_get_pref_value* -key *Key* [-category *Category*]

```
Key             String
Category        String
```

Arguments

```
-key Key
```

Key specifies the key to retrieve the value from.

Feedback

`-category Category`

Category specifies the category to search for the key. If not specified, a default category will be used.

Description

This command retrieves the value of a preference key from the specified key name and category.

Examples

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false. The command *gui_get_pref_value* is then used to retrieve the value of the preference key just created.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type bool
 -value false
set x [gui_get_pref_value -category my_category -key some_key]
```

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_get_pref_value_type

Get the data type of the value of a preference key.

Syntax

string *gui_get_pref_value_type* -key *Key* [-category *Category*]

Feedback

```
Key             String
Category        String
```

Arguments

`-key Key`

Key specifies the key to retrieve the value type from.

`-category Category`

Category specifies the category to search for the key. If not specified, a default category will be used for searching the key.

Description

This command retrieves the data type of the value of a preference key from the specified key name and category.

Examples

The following example create a user-defined category *some_category* for storing preferences and then create an integer key *some_key* under that category with the initial value of 200. The command *gui_get_pref_value_type* is then used to retrieve the data type of the value of the preference key just created.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type
 integer -value 200
set x [gui_get_pref_value_type -category my_category -key some_key]
{comment: x should be equal to "integer" }
```

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

Feedback

gui_get_presets

Gets list of preset name for object specified by category

Syntax

```
string gui_get_presets
-category category
[-system]
[-shared]
```

Data Types

```
category    string
```

Arguments

`-category category`

Specifies the name of the category that the presets be retrieved from.

`-system`

Return system presets. This option is mutually exclusive with the -shared option.

`-shared`

Return shared presets. This option is mutually exclusive with the -system option.

Description

This command retrieves the list of presets for a category.

Examples

The following example gets the system presets for the Layout.

```
prompt> gui_get_presets -category Layout -system
```

See Also

- gui_set_preset

gui_get_region

Returns the coordinates of the current region rectangle or rectilinear polygon.

Syntax

```
list gui_get_region
```

Feedback

Description

This command returns the coordinates of the rectangle or rectilinear polygon that defines the current region in layout region mode.

Note that the preferred command for typical extension writing is usually *gui_set_layout_user_command* instead of *gui_get_region* because *gui_set_layout_user_command* has snapping, mouse up drag, cancel, and user prompt capabilities.

Examples

```
prompt> gui_get_region
{{-1743.945 18.357} {42.833 1523.657}}
```

See Also

- gui_set_region
- gui_set_layout_user_command

---

gui_get_setting

Gets a setting on the specified window.

Syntax

string *gui_get_setting*

```
-window WindowID
{ -setting Setting | -list }

WindowID         String
Setting         String
```

Arguments

`-window WindowID`

WindowID specifies the window to get the setting

`-setting Setting`

Setting specifies the setting to get

`-list`

Specifies that the list of available settings needs to be returned

Feedback

Description

Gets the list of available settings on the specified window. Windows are views or top level windows. The setting is a property of the window. The returned value is a string which has the type of the setting being read. Some windows might not have this function implemented.

Examples

```
shell> gui_get_setting -window Layout.1 -setting showCell
1

shell> gui_get_setting -window Layout.1 -setting viewLevel
2
```

See Also

- gui_set_setting

---

gui_get_task_list

Lists all the available task names.

Syntax

string *gui_get_task_list*

Description

The gui_get_task_list command queries the names of all available tasks.

Description

prompt> *gui_get_task_list* all {Block Implementation} {Design Planning}

See Also

- gui_create_task
- gui_set_task_list
- gui_set_current_task
- gui_get_current_task

---

gui_get_task_page

Return the name of the task page for the given task item.

Feedback

Syntax

string *gui_get_task_page*

Syntax

gui_get_task_page

```
-task           TaskName
-item           ItemName

TaskName          String
ItemName          String
```

Arguments

`-task` *TaskName*

TaskName specifies the name of the task.

`-item` *ItemName*

ItemName specifies the hierarchical name of the task item for which to get the page name",

Description

This command returns the name of the task page for the task item identified by the option values. This is useful when you want to create a new task item re-using a task page that is invoked by an existing task item.

Examples

To reuse the task page from the built-in task item "Pin Assignment->Pin Placement" in the task "Hierarchical Design" in your own task flow, you can do the following:

```
prompt> set pageName [gui_get_task_page -task "Hierarchical Design" \
                        -item "Pin Assignment->Pin Placement"]
prompt> gui_create_task_item -task myTask -name myItemName \
                        -page $pageName
```

See Also

- gui_create_task_item
- gui_set_current_task
- gui_get_task_list

Feedback

gui_get_toolbar_names

Return a Tcl list of the names of all the toolbars that have been created with the *gui_create_toolbar* command.

Syntax

void *gui_get_toolbar_names* [-window *WindowId*]

```
WindowId        String
```

Arguments

`-window WindowId`

Return toolbars defined in this window. If omitted, return toolbars defined in the active window.

Description

Return a Tcl list of the names of all the toolbars that have been created with gui_create_toolbar command for the specified window.

See Also

- gui_create_toolbar
- gui_create_toolbar_item
- gui_delete_toolbar
- gui_delete_toolbar_item
- gui_hide_toolbar
- gui_show_toolbar

gui_get_user_units

Gets the units for GUI.

Syntax

```
status gui_get_user_units
-type unit_type
[-numeric]
```

Data Types

```
unit_type          string
```

Feedback

Arguments

`-type unit_type`

Specifies the type of unit to get. The valid value is: "power".

`-numeric`

If the -numeric option is specified, the command returns a number representing the unit value in terms of the base unit for that quantity type. The base unit for power is W (watt).

Description

This command returns a string representing the GUI unit for a quantity type.

Examples

The following examples get the unit of power.

```
prompt> gui_get_user_units -type power
1kW
prompt> gui_get_user_units -type power -numeric
1000
```

See Also

- gui_set_user_units

---

gui_get_utable

Get various state about User Table(s) [default: return list of all table names].

Syntax

string *gui_get_utable*

```
-name           Table_Name
-has_name
-headers
-tag
-window         Window_name
-values         Column_Name
-bus_values     Column_Name
-rows
-parent
-filter
-columns

Table_Name      String
Window_Name     String
Column_Name     String
```

Feedback

Arguments

`-name Table_Name`

A table name argument used with other argument switches below.

`-has_name`

Check if given table name given by the '-name' argument switch exist and return '1', else return '0'.

`-headers`

Get the table column headers for given table argument.

`-tag`

Get the table MetaData tag string for given table argument.

`-window Window_Name`

A UserTable window name argument used with other arguments below. If used alone it will return the current table name of the given window name.

`-values Column_Name`

Get the selected table cell value(s) in the given column for the given window argument. *Note*: Any column value(s) in the row of the selected cell(s) can be retrieved.

`-bus_values Column_Name`

Just like the '-values' option, except if the columns contain compressed bus values like 'bus*[*]*', then a list of values will be returned that match the glob expression.

`-bus_values Column_Name`

Get the selected table cell bus values in the given column for the given window argument. If the table cell value is a compressed bussed value then all values are returned for that compressed value.

`-columns`

Get the selected table column name(s) for the given window argument.

`-rows`

Get the selected table row number(s) for the given window argument.

`-parent`

Get the parent table name for the given table name argument if it exists.

`-filter`

Get the filter expression for the current table shown in the given window name.

Feedback

Description

This command allows you to get all current loaded UserTable names.
Other options allow you to get various other table state.

Examples

The following example check to see if the UserTable 'my_table' is currently loaded.

```
shell> if { [gui_get_utable -name my_table -has_name] } {
          echo "yes my_table is loaded"
       }
```

The following example looks for the 'full_name' column in the current active UserTable
and prints all the selected values in that column.
Note: If the table was a compressed table then using the option '-bus_values'
will return all the values for the compressed value.

```
shell> # Get the current active UserTable window:
       set win [gui_get_current_window -mru -type UserTable]
       # Get the table name in that window:
       set table [gui_get_utable -window $win]

       if { $table != {} } {
         # Get the selected column names in that table:
         set columns [gui_get_utable -window $win -columns]
         # If the column name 'full_name' is selected then list the values:
         if { [lsearch -exact $columns full_name] != -1} {
           set values [gui_get_utable -window $win -values full_name]
           foreach v $values {
             echo "value: $v"
           }
         }
       }
```

See Also

- gui_append_utable
- gui_close_utable
- gui_change_selection_utable
- gui_create_utable
- gui_export_utable
- gui_foreach_utable_row

Feedback

- gui_import_utable
- gui_open_utable
- gui_set_utable
- gui_set_utable_meta
- gui_set_utable_values
- gui_show_utable
- gui_write_utable

---

gui_get_vm

Retrieves attributes for a specified visual mode.

Syntax

string *gui_get_vm*

```
-name identifier
[-buckets]
[-title]
[-help_topic]
[-infotip]
[-discrete]
[-icon_file]
[-update_cmd]
[-netfilter]
[-float]
[-top_exaggeration]
[-mid_exaggeration]
[-bot_exaggeration]
[-show_only_pins_of_nets]
[-enable_bucket_delete]
```

Data Types

```
identifier     string
```

Arguments

`-name` *identifier*

Specifies the name of the visual mode to be queried.

`-buckets`

Returns a list of bucket names. The bucket names are retrieved in rendering order, starting from the first to the last bucket rendered.

`-title`

Returns the title string.

`-help_topic`

Returns the help topic string.

`-infotip`

Returns the InfoTip string.

`-discrete`

Returns a true or false value that indicates whether the buckets of this visual mode are discrete or continuous. Only discrete buckets can be reordered. This attribute is set when this visual mode was created, and cannot be changed.

`-icon_file`

Returns the gui menu icon file.

`-update_cmd`

Returns the Tcl command that is executed when the visual mode is accessed. A typical use for this option is to update the state of the visual mode if its contents are likely to change under certain circumstances.

`-netfilter`

Returns net connection filtering.

`-float`

Returns a true or false value indicating whether the visual mode buckets have a floating point range.

`-top_exaggeration`

Returns the top bucket exaggeration.

`-mid_exaggeration`

Returns the middle bucket exaggeration.

`-bot_exaggeration`

Returns the bottom bucket exaggeration.

`-show_only_pins_of_nets`

Returns a true or false value that indicates whether the layout shows only pins of net objects in buckets.

Feedback

`-enable_bucket_delete`

Returns a true or false value indicating whether the visual mode buckets can be deleted from context menu.

Description

The *gui_set_vm* command queries the option values of an existing visual mode.

Examples

To retrieve names of the buckets in the visual mode named "mycoloring1", use the following command:

```
prompt>gui_get_vm -name mycoloring1 -buckets
```

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vmbucket
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vm
- gui_set_vmbucket
- gui_update_vm

---

gui_get_vmbucket

Get attributes for Visual Mode Bucket

Syntax

string *gui_get_vmbucket* -vmname *mode_identifier*

-name *bucket_identifier* [-infotip] [-netfilter] [-color] [-pattern] [-exaggeration] [-number] [-maxval] [-minval] [-visible] [-title] [-collection] [-objcount] [-gui_object]

```
string mode_identifier
string bucket_identifier
```

Feedback

Arguments

`-vmname` *mode_idnetifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

`-name` *bucket_identifier*

Specifies the name of the visual mode bucket to be queried. To see the list of buckets associated with a specific visual mode use the gui_get_vm command with the -buckets option.

`-infotip`

Return infotip string.

`-netfilter` *net_filter*

Return net connection filtering.

`-color`

Return bucket rendering color.

`-pattern`

Return bucket rendering fill pattern.

`-exaggeration`

Return bucket min pixel exaggeration value.

`-number`

Return bucket display number.

`-maxval`

Return bucket maximum value.

`-minval`

Return bucket minimum value.

`-visible`

Return visibility.

`-title`

Return bucket title.

`-collection`

Return object collection.

Feedback

`-objcount`

Return the number of objects in the bucket.

`-gui_object`

Return gui object collection.

You can use this option only if you also use the *-collection* option.

Description

The gui_get_vmbucket command queries an instance of a visual mode bucket for the specified option values. The values are returned in a list of option-value pairs.

Examples

Query the current color for the bucket named "cat1" in the visual mode "foo":

shell> *gui_get_vmbucket -vmname foo -name cat1 -color*

Query the current color and pattern for the bucket named "cat1" in the visual mode "foo":

shell> *gui_get_vmbucket -vmname foo -name cat1 -color -pattern*

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm
- gui_set_vmbucket
- gui_list_vm
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vm
- gui_update_vm

---

gui_get_window_ids

Get a list of window ids

Feedback

Syntax

```
ids gui_get_window_ids
[ -parent window_id ]
[ -type window_type ]
```

Data Types

```
window_id       string
window_type     string
```

Arguments

`-parent window_id`

Specifies the toplevel window id to get child view window ids from.

`-type window_type`

Specifies that the windows id returned should be restricted to those of the specified type.

Note: as a type can be only be associated with either a toplevel window or a child view window (not both) the type implies whether toplevel or child view window ids are returned.

If this option is not specified then either all toplevel window ids are returned (*-parent* option not specified) or all child view window ids of a specified toplevel window id are returned (*-parent* option specified).

Returns

ids

Returned list of window ids.

Description

This command gets a list of the ids of all toplevel windows, all toplevel windows of a specified type, all child view windows of a specified type, or all child view windows of a specified type in a specified toplevel window.

Examples

The following example will create a toplevel window and a child layout window. Then this command will be used to retrieve the ids of the existing windows.

```
shell> set top [gui_create_window -type TopLevel]
shell> set cons [gui_create_window -type Layout -parent $top]
shell> set ids [gui_get_window_ids]   # return list containing values of
 $top and value of $cons.
shell> set childs [gui_get_window_ids -parent $top]   # return value of
 $cons.
```

Feedback

```
shell> set layout_instances [gui_get_window_ids -type Layout] # return
 value of $cons.
```

See Also

- gui_close_window
- gui_exist_window
- gui_show_window
- gui_get_window_types
- gui_create_window
- gui_set_active_window
- gui_get_current_window

---

gui_get_window_pref_categories

Get list of preference categories for object specified by window

Syntax

```
categories gui_get_window_pref_categories
{ -window window_id | -window_type window_type }
```

Data Types

```
window_id       string
window_type     string
```

Arguments

`-window window_id`

Specifies the name of the window that the preference categories will be retrieved from. This option is mutually exclusive with the -window_type option.

`-window_type window_type`

Specifies the name of the window type that instances of the type will have the preference categories be retrieved from. This option is mutually exclusive with the -window option.

Returns

categories

The returned preference categories.

Feedback

Description

This command retrieves the list of preference categories for a window or window type.

Examples

The following example gets the preference categories for the TopLevel.1 window.

```
shell> ser cats [gui_get_window_pref_categories -window TopLevel.1]
```

See Also

- gui_set_window_pref_key
- gui_get_window_pref_value
- gui_get_window_pref_keys
- gui_create_pref_category
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_get_window_pref_keys

Get list of preference categories for object specified by window

Syntax

```
categories gui_get_window_pref_keys
{ -window window_id | -window_type window_type }
[ -category category ]
```

Data Types

```
window_id       string
window_type     string
category        string
```

Feedback

Arguments

`-window window_id`

Specifies the name of the window that the preference categories will be retrieved from. This option is mutually exclusive with the -window_type option.

`-window_type window_type`

Specifies the name of the window type that instances of the type will have the preference categories be retrieved from. This option is mutually exclusive with the -window option.

`-category category`

Specifies the category that the key belongs to. If not specified, a default category will be assigned.

Returns

categories

The returned preference categories.

Description

This command retrieves the list of preference categories for a window or window type.

Examples

The following example gets the preference categories for the TopLevel.1 window.

```
shell> ser cats [gui_get_window_pref_keys -window TopLevel.1]
```

See Also

- gui_set_window_pref_key
- gui_get_window_pref_value
- gui_get_window_pref_categories
- gui_create_pref_category
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories

Feedback

- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_get_window_pref_value

Get preference value for object specified by window or window type

Syntax

```
value gui_get_window_pref_value
{ -window window_id | -window_type window_type }
[ -category category ]
-key key
```

Data Types

```
window_id       string
window_type     string
category        string
key             string
```

Arguments

`-window window_id`

Specifies the name of the window that the preference will be retrieved from. This option is mutually exclusive with the -window_type option.

`-window_type window_type`

Specifies the name of the window type that instances of the type will have the preference be retrieved from. This option is mutually exclusive with the -window option.

`-category category`

Specifies the category to search for the key. If not specified, a default category will be used.

`-key key`

Specifies the key to retrieve the value from.

Returns

value

The value of the key.

Feedback

Description

This command retrieves the value of a preference key from the specified key name and category of the specified window or window type. If a preference is set on a window type, then all instances of that type will inherits the preference, so the value can be set on a type and retrieved from any instance of that type (or derived from that type).

Examples

The following example creates a preference for a window and then uses the *gui_get_window_pref_value* command to retrieve the value for the preference.

```
shell> gui_set_window_pref_key -window Console.1 -key test -value_type
 integer -value 201
shell> set x [gui_get_window_pref_value -window Console.1 -key test]
shell> if {$x == 201} { echo "success" }
```

See Also

- gui_set_window_pref_key
- gui_get_window_pref_categories
- gui_get_window_pref_keys
- gui_create_pref_category
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_get_window_presets

Gets list of preset name for object specified by window_type

Feedback

Syntax

```
string gui_get_window_presets
-window_type window_type
[-system]
[-shared]
```

Data Types

window_type	string

Arguments

`-window_type` *window_type*

Specifies the name of the window type that the presets be retrieved from.

`-system`

Return system presets. This option is mutually exclusive with the -shared option.

`-shared`

Return shared presets. This option is mutually exclusive with the -system option.

Description

This command retrieves the list of presets for a window type. This command has been deprecated, please use gui_get_presets instead.

Examples

The following example gets the system presets for the Layout.

```
prompt> gui_get_window_presets -window_type Layout -system
```

See Also

- gui_get_presets
- gui_set_preset
- gui_set_window_preset

---

gui_get_window_toolbar_items

Get window toolbar items

Syntax

```
status gui_get_window_toolbar_items
[ -window window_id ]
```

Feedback

```
[ -toolbar string ]
[ { -group string | -item string } ]
```

Data Types

window_id string

Arguments

-window *window_id*

Specifies the window name id.

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

-toolbar *string*

Specifies the name of the toolbar to return items for.

If the string '?' is specified for the type, the command will return a list of allowed set names.

-group *string*

Specifies the name of the group to return.

If the string '?' is specified for the type, the command will return a list of allowed item groups.

-item *string*

Specifies the name of the item to return.

If the string '?' is specified for the type, the command will return a list of allowed item names.

Description

This command returns a collection of existing window toolbar items.

If no options are specified then all items are returned, if a set is specified then all items in the set are returned, if an item is specified then just that item (if it exists) is returned.

Examples

The following example will creates a button in a window toolbar and gets a collection for it.

```
shell> proc init_my_toolbar { args } {
shell>   set button [gui_create_window_toolbar_item -name button -type
 button]
shell> }
shell> gui_create_window_toolbar_type -type my_toolbar -name "My Toolbar"
 \\
```

Feedback

```
shell>   -view_types Charts -command init_my_toolbar
shell> set window [gui_create_window -type TopLevel]
shell> set charts [gui_create_window -type Charts]
shell> gui_show_window_toolbar -window $charts
shell> set button [gui_get_window_toolbar_items -window $charts -toolbar
 my_toolbar -item button]
```

See Also

- gui_create_window_toolbar_type
- gui_create_window_toolbar_item
- gui_create_window_toolbar_group
- gui_create_window

---

gui_get_window_types

Get a list of window types

Syntax

```
types gui_get_window_types
[ -type token ]
```

Data Types

```
token    string
```

Arguments

`-type window_types`

Specifies whether to return toplevel windows ('toplevel') or child view windows ('child').

If toplevel window types are specified then the first type is guaranteed to be the default toplevel window type.

Returns

types

Returned list of window types.

Description

This command returns a list of existing window types. Instances of these types can be instantiated with the *gui_create_window* command. The returned list can be restricted to just toplevel windows or child view windows by specifying the *-type* option.

Feedback

Examples

The following examples illustrate some usages of the command.

```
shell> gui_get_window_types
shell> gui_get_window_types -type toplevel
shell> gui_get_window_types -type child
```

See Also

- gui_close_window
- gui_exist_window
- gui_show_window
- gui_create_window
- gui_get_window_ids
- gui_set_active_window
- gui_get_current_window

---

gui_group_charts_plots

Group plots in charts view

Syntax

```
status gui_group_charts_plots
-view view_name
[ -x1x2 ]
[ -y1y2 ]
[ -overlay ]
[ plots ]
```

Data Types

```
view_name     string
plot_id_list  list
```

Arguments

`-view view_name`

Charts View to group.

`-x1x2`

This option specifies that two plots are connected such that they have a shared y axis range but two independent x axes ranges.

If the *-overlay* option is used then the two plots are overlaid and the X axis of the first plot is placed on the bottom and the X axis of the second plot is placed on the top. Note: To ensure the plots are overlaid correctly the two plots are forced to the same area of the view (usually the whole view).

If the *-overlay* option is not used then the two plots are not overlaid and can be placed anywhere in the view. For best results horizontal stacking is recommended so the shared y axes are placed next to each other.

There must be exactly two plots specified to the *-plots* option.

This option cannot be combined with the *-y1y2* option.

`-y1y2`

This option specifies that two plots are connected such that they have a shared x axis range but two independent y axes ranges.

If the *-overlay* option is used then the two plots are overlaid and the Y axis of the first plot is placed on the bottom and the Y axis of the second plot is placed on the top. Note: To ensure the plots are overlaid correctly the two plots are forced to the same area of the view (usually the whole view).

If the *-overlay* option is not used then the two plots are not overlaid and can be placed anywhere in the view. For best results vertical stacking is recommended so the shared x axes are placed next to each other.

There must be exactly two plots specified to the *-plots* option.

This option cannot be combined with the *-x1x2* option.

`-overlay`

This option specifies that the plots have shared x and/or y ranges and are drawn on top of each other so they can share a single key and title.

If the *-x1x2* or *-y1y2* option are specified then only the x or y axis range is shared (See descriptions of these options above).

If neither of the *-x1x2* or *-y1y2* options are specified then both the x and y axis ranges are shared. In this case two or more plots can be specified.

`plots`

List of plots to group.

For *x1x2* or *y1y2* options this must be two plots. For *-overlay* option, without the *x1x2* and *y1y2* options, this must be two or more plots.

Description

Group related plots in a view.

Feedback

Grouped plots can share x or y axis ranges and can be overlaid.

Sharing x and/or y axis is useful when you want to compare data across two plots. Overlay is useful when you want to display multiple plot types with the same data e.g. a combined barchart and xy plot (line plot).

Examples

The following example groups two plots so they are overlaid and share a common x axes.

```
shell> gui_group_charts_plots -view $view -y1y2 -overlay -plots [list
 $plot1 $plot2]
```

See Also

- gui_create_charts_plot

---

gui_hide_palette

Hides the specified palette.

Syntax

```
status gui_hide_palette
{ -name palette_id | -type palette_type | -all }
[ -parent window_id ]
```

Data Types

```
palette_id    string
palette_type  string
window_id     string
```

Arguments

`-name palette_id`

Specifies the palette name id or title.

If the *-parent* option is specified, only palettes in the specified parent toplevel window are hidden. If the *-parent* option is not specified, all palettes of this name in all toplevel windows are hidden.

This option precludes the *-type* and *-all* options.

`-type palette_type`

Specifies the palette type id. Any matching palette of this type will be hidden.

If the *-parent* option is specified, only palettes in the specified parent toplevel window are hidden. If the *-parent* option is not specified, all palettes of this type in all toplevel windows are hidden.

Feedback

This option precludes the *-name* and *-all* options.

`-all`

Hides all palettes.

If the *-parent* option is specified, only palettes in the specified parent toplevel window are hidden. If the *-parent* option is not specified, all palettes in all toplevel windows are hidden.

This option precludes the *-name* and *-type* options.

`-parent window_id`

Specifies the parent toplevel window to hide the palette type in.

Note: this option is only applicable if the *-type* option is specified.

Description

This command hides the specified palettes.

Examples

The following example hides all Layer palettes:

```
shell> gui_hide_palette -type Layout
```

See Also

- gui_show_palette

---

gui_hide_toolbar

Hides the specified toolbar or all the toolbars in the specified window or for a window type.

Syntax

void *gui_hide_toolbar*

```
-toolbar tool_bar_name | -all
[-window window_id]
[-window_type window_type]
```

Data Types

```
tool_bar_name     string
window_id         string
window_type       string list
```

Feedback

Arguments

`-toolbar` *`tool_bar_name`*

Specifies the toolbar you want to hide. The *-toolbar* option and the *-all* option are mutually exclusive.

`-all`

Hides all the toolbars in the specified window. The *-all* option and the *-toolbar* option are mutually exclusive.

`-window` *`window_id`*

Specifies the window in which you want to hide the specified toolbar or all toolbars. This option is mutually exclusive with the *-window_type* option. If both *-window* and *-window_type* options are omitted, then by default the tool hides the toolbar or toolbars in the active window.

`-window_type` *`window_type`*

Specifies the window types in which you want to hide the specified toolbar or all toolbars. All existing windows of the given types will be affected. Toolbar visibility may be set on a window type before a window of the type exists. This option is mutually exclusive with the *-window* option. If both *-window* and *-window_type* options are omitted, then by default the tool hides the toolbar or toolbars in the active window.

Description

This command hides either all the toolbars or the toolbar with the specified name in the window with the specified window ID, or in all windows of the given window type.

Toolbar visibility set by this command will override any visibility setting that has been saved and restored from previous sessions. The last toolbar visibility set by this command or by *gui_show_toolbar* will be saved and restored on next invocation of the tool if save and restore has not been disabled.

Examples

The following example hides the File toolbar in the active window:

```
prompt> gui_hide_toolbar -toolbar File
```

The following example hides all the toolbars in the layout window named LayoutWindow.1:

```
prompt> gui_hide_toolbar -all -window LayoutWindow.1
```

The following example hides the File toolbar in all TimingWindow type windows:

```
prompt> gui_hide_toolbar -all -window_type TimingWindow
```

Feedback

The following example hides all the toolbars in all window types:

```
prompt> gui_hide_toolbar -all -window_type [gui_get_window_types -type
 toplevel]
```

See Also

- gui_create_toolbar
- gui_delete_toolbar
- gui_create_toolbar_item
- gui_delete_toolbar_item
- gui_show_toolbar
- gui_get_toolbar_names
- gui_get_window_types

---

gui_hide_window_toolbar

Hide toolbar in view

Syntax

```
status gui_hide_window_toolbar
-window window_id
```

Data Types

```
window_id  string
```

Arguments

```
-window window_id
```

Specifies the view name id.

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

Description

This command hides the specified view's toolbar.

Examples

The following example will show a Charts View, show and hide the toolbar.

```
shell> set top [gui_create_window -type TopLevel]
shell> set charts [gui_create_window -type Charts -parent $top]
```

Feedback

```
shell> gui_show_window_toolbar -window $charts
shell> gui_hide_window_toolbar -window $charts
```

See Also

- gui_create_window
- gui_show_window_toolbar

---

gui_import_utable

Create a UserTable and import records from a value file or a report.

Syntax

string *gui_import_utable*

```
-file            File_Name [-tsv_mode] [-ssv_mode]
[-name           Table_Name]
[-tsv_mode]
[-ssv_mode]
[-fold]
[-report_type    Report_Type_Name]
[-meta_obj       MetaObj_Name]
[-replace]

Table_Name            String
File_Name             String
Report_Type_Name      String
MetaObj_Name          String
```

Arguments

`-file File_Name`

The name of the CSV value file from which to import and fill the user table with.

`-name Table_Name`

The name for the user table created. By default, the name takes the name of the file minus the extension.

`-tsv_mode`

This flag changes the default delimiter from the comma char to the tab char.

`-ssv_mode`

This flag changes the default delimiter from the comma char to the semicolon char.

`-fold`

This flag causes extra row data past the number of columns specified to fold into the last column. The default is to discard extra data.

`-report_type`

This flag causes the reader to interpret the file as a report file and convert it into a user table. *Note:* Currently only the 'report_constraint -all_violators' report can be converted.

`-meta_obj` *`MetaObj_Name`*

Optional MetaData object name used to define table meta data. See the command gui_set_utable_meta for more information.

`-replace`

Optional flag that will replace the given table name, if it already exists, with the imported table data.

Description

```
This command creates a new UserTable in memory and imports the data from
 a CSV/TSV/SSV
value file. The name of the table will take the base name of the file
 unless a name is
provided.

The first line of the file is assumed to be a header line that defines
 the column names.
In addition, column names can have postfixes that describe the data
 format of that column.

Supported column name postfixes are:


  :string
  :int
  :double
  :cell
  :net
  :port
  :pin
  :point
  :endpoint
  :file
```

Examples

```
The following example creates a new UserTable called 'data' and imports
 the values from
the CSV file 'data.csv'.
```

Feedback

```
shell> gui_import_utable -file data.csv

The following example creates a new UserTable, names it 'my_table' called
 'data' and
imports the values from the CSV file 'data.csv'. It also instructs the
 import to fold
any extra data found in the row into the last column defined.

shell> gui_import_utable -file data.csv -name my_table -fold
```

See Also

- gui_append_utable
- gui_change_selection_utable
- gui_close_utable
- gui_create_utable
- gui_export_utable
- gui_get_utable
- gui_set_utable_meta
- gui_open_utable
- gui_show_utable
- gui_write_utable

---

gui_list_attrgroups

Lists attribute group information for a specified object type or all object types.

Syntax

```
status gui_list_attrgroups
-class design_object
-name name
[-all]
[-tcl]
[-full]
[-attr_list]
```

Data Types

```
design_object      string
name               string
```

Feedback

Arguments

`-class` *design_object*

Specifies the type of design object.

`-name` *name*

Specifies the name of the attribute group to be listed for the specified object type or for all object types.

`-all`

Lists the attribute groups for all object types. This option and the -class option are mutually exclusive.

`-tcl`

Lists the attribute group information in a tool command language (Tcl) format that is suitable for Tcl processing.

`-full`

Specifies the full Tcl output format, which reports all possible group information.

`-attr_list`

Lists attributes only.

Description

The *gui_list_attrgroups* command lists attribute group information for the specified object type or for all object types. Use the *-tcl* option to get results that are suitable for Tcl processing.

Examples

```
prompt> gui_list_attrgroups -class Cell -tcl
"Hierarchy" "Edit"

prompt> gui_list_attrgroups -class Cell -name "Hierarchy" -tcl -attr_list
{Name} {Hierarchical} {Owning Cell} {isPhysConstraint} {Hard Macro} ...
 {FullName}

prompt> gui_list_attrgroups -all -tcl
Cell {  "Hierarchy" "Edit" }  Net {  "Timing" }  Pin {  "Timing" }

prompt> gui_list_attrgroups -class Cell -tcl -attr_list
{ "Hierarchy" { {Name} {Hierarchical} {Owning Cell} ...{FullName} } }
{ "Edit" { {Name} {isPhysConstraint} {Orientation} ... {FullName} } }

prompt> gui_list_attrgroups -class Cell
Cell, "Hierarchy", { {Name} {Hierarchical} {Owning Cell}
 {isPhysConstraint} ... }  ;
```

Feedback

```
Cell, "Edit", { {Name} {isPhysConstraint} {Orientation} {Location} ... }
  ;
The fields displayed above are in the following order:
Class, Group Name, Attributes list;
```

See Also

- gui_delete_attrgroup
- gui_list_attrgroups
- gui_update_attrgroup

---

gui_list_category_rules

Lists all category rules except built-in rules by default.

Syntax

list *gui_list_category_rules*

```
[-all | -names rule_names_list]
[-format script | tcl_list]
```

Data Types

rule_names_list list

Arguments

`-all`

Lists all category rules, including built-in rules.

This option and the *-names* option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

`-names` *rule_names_list*

Specifies the names of the category rules to be listed.

This option and the *-all* option are mutually exclusive. If you do not specify either of these options, the command lists all category rules except the built-in rules.

`-format script | tcl_list`

Specifies the output format in which the category rules are listed. The default output format is *script*.

When the *script* format is used, the category rules are listed as a Tool Command Language (Tcl) script of *gui_create_category_rule* commands that you can

replay. In this case, the rules are streamed to the output stream. You can redirect the output by using the *redirect* command, for example, if you want to capture the script output in a file to use again later. When the *script* format is used, the command result is an empty string.

If the *tcl_list* format is specified, the category rules are listed in an "array set compatible" Tcl list format. In this case, the command result is a Tcl list.

Description

This command lists category rules that have already been created by the *gui_create_category_rule* command during the current run of the tool.

When you run this command without specifying either the *-all* option or the *-names* option, the command lists all category rules except built-in rules. Specify the *-all* option to list all category rules, including the built-in rules. Specify the *-names* option to list individual rules by name.

Examples

The following example lists all category rules, including the built-in rules:

```
prompt> gui_list_category_rules -all
gui_create_category_rule  -name Pathgroups  -builtin \\
  -category <path_group.full_name>
gui_create_category_rule  -name Session  -builtin \\
  -category <session_name>
gui_create_category_rule  -name {Path Type}  -builtin \\
  -category <path_type>
  ...
  ...
```

The following example lists all category rules except the built-in rules:

```
prompt> gui_create_category_rule  -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule  -name rule2 -category
 <path_group.full_name>
rule2
prompt> gui_list_category_rules
gui_create_category_rule  -name rule1 \\
  -category <session_name>
gui_create_category_rule  -name rule2 \\
  -category <path_group.full_name>
```

The following example redirects the script output to a file that you can source during a subsequent run of the tool:

```
prompt> redirect -file /remote/dir1/rules.tcl {gui_list_category_rules}
prompt>
```

The following example uses the *-names* option to list only the rules named rule1 and rule3:

```
prompt> gui_create_category_rule  -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule  -name rule2 -category
 <path_group.full_name>
rule2
prompt> gui_create_category_rule  -name rule3 -category
 <startpoint.full_name>
rule3
prompt> gui_list_category_rules -names {rule1 rule3}
gui_create_category_rule  -name rule1 \\
  -category <session_name>
gui_create_category_rule  -name rule3 \\
  -category <startpoint.full_name>
```

The following example uses the *tcl_list* output format to query the category specification of the rule named rule1:

```
prompt> gui_create_category_rule  -name rule1 -category <session_name>
rule1
prompt> gui_create_category_rule  -name rule2 -category
 <path_group.full_name>
rule2
prompt> array set rules [gui_list_category_rules -format tcl_list]
Information: Defining new variable 'rules'. (CMD-041)
prompt> array set rule1_options $rules(rule1)
Information: Defining new variable 'rule1_options'. (CMD-041)
prompt> set rule1_cat_spec $rule1_options(category)
Information: Defining new variable 'rule1_cat_spec'. (CMD-041)
<session_name>
```

See Also

- gui_create_category_rule
- gui_remove_category_rules

---

gui_list_cell_block_marks

Lists the cell names and block mark values for cells that are marked in the design.

Syntax

list *gui_list_cell_block_marks*

Arguments

This command has no arguments

Feedback

Description

This command lists the cell names and block mark values for all cells that are marked in the design. The listing is sorted by the full names of the marked cells.

Examples

The following example sets block marks on two hierarchical cell instances, and then lists the names of all the marked cells and their block marks:

```
prompt> gui_remove_cell_block_marks -all
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_set_cell_block_marks I_TOP/I_REG_FILE  REG_FILE
prompt> gui_list_cell_block_marks
I_TOP/I_ALU    ALU
I_TOP/I_REG_FILE    REG_FILE
```

See Also

- gui_set_cell_block_marks
- gui_get_cell_block_marks
- gui_remove_cell_block_marks

---

gui_list_vm

List Current Visual Modes

Syntax

string *gui_list_vm*

Description

The gui_list_vm command will return a list of existing visual modes.

Examples

To get the names of all available visual modes, use the following command:

shell> *gui_list_vm*

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm

Feedback

- gui_set_vm
- gui_get_vmbucket
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vmbucket
- gui_update_vm

gui_load_cell_density_mm

Loads the data for cell density map mode.

Syntax

```
status  gui_load_cell_density_mm
[-area area]
```

Data Types

area	list

Arguments

`-area` *area*

Analyzes area for cell density. If area is not given, the default area is whole design area.

Description

The command generates the map mode view of cell density in a given area. Each grid is colored based on the percentage of occupied area of cells in the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example shows cell density in the specified area.

```
prompt> gui_load_cell_density_mm -area \\
{284.820 390.670 1202.790 954.120}
```

gui_load_cell_power_density_mm

Loads the data for cell power density map mode.

Feedback

Syntax

```
gui_load_cell_power_density_mm
[-update_power]
```

Arguments

`-update_power`

Enables power analysis and updates the power data if it is not up-to-date.

Description

The *gui_load_cell_power_density_mm* command generates map mode view of cell power density. The grids are colored based on the total power generated from cells within the grid.

To generate the power data for cells, 1. Power analysis must be enabled ahead of time by setting the *power_enable_analysis* variable to *true*. 2. You have to use the *update_power* command to calculate the latest power information. Alternatively, you can also use *-update_power* option to set the variable and perform the fBgui_load_cell_power_density_mm *command.*

Examples

The following example shows cell power density map.

```
prompt> gui_load_cell_power_density_mm -update_power
```

See Also

- update_power
- power_enable_analysis

---

gui_load_pin_density_mm

Loads the data for pin density map mode.

Syntax

status *gui_load_pin_density_mm*

`[-area area]`

Data Types

area list

Feedback

Arguments

`-area area`

Area to be analyzed for pin density. By default, the command uses the entire design area.

Description

The command generates map mode view of pin density in a given area. The grids are colored based on the number of pins within the grid.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example shows pin density in the specified area.

```
prompt> gui_load_pin_density_mm -area \\
  {284.820 390.670 1202.790 954.120}
```

---

gui_log_performance

Controls the start and stop of a performance data logging process and outputs into a file.

Syntax

```
string gui_log_performance
[-start file_name|-stop]
[-compress]
[-quiet]
```

Data Types

file_name	string

Arguments

`-start file_name`

This option is mutually exclusive with -stop. The argument is the path name of the file to which the log is output. The option begins logging performance data. If the file already exists, the command overwrites the existing file.

`-stop`

This option is mutually exclusive with -start. This option stops logging performance data. After stopped, the log cannot be reopened for appending.

Feedback

`-compress`

If given, the log is saved as a compressed file. This option must be combined with -start.

`-quiet`

If given, this option causes the command to ignore error conditions.

Description

This command controls the start and stop of a performance data logging process. The performance data is logged before and after executing the commands which are required. The data includes cpu, memory and time. After successfully start a log, the command returns the path name with extension .log or .log.gz if not given in *file_name*.

Examples

```
Start logging and save the log to a file without compressing.

prompt> gui_log_performance -start ./test
/an/absolute/path/.test.log

Start logging and save the log to a compressed file.

prompt> gui_log_performance -start ./test -compress
/an/absolute/path/.test.log.gz

Stop logging

prompt> gui_log_performance -stop
```

---

gui_merge_utable

Merge 2 (loaded) tables via the given join column(s).

Syntax

string *gui_merge_utable*

```
-name             Table_Name
-merge_table      Table_Name
-join_columns     Column_Names
[-output_table    Table_Name]
[-columns         Column_Names]

Table_Name       String
Column_Names     List of Strings
```

Feedback

Arguments

`-name` *`Table_Name`*

A primary table name to merge into unless an output table name is also given.

`-merge_table` *`Table_Name`*

The merge table name. This table must have the same 'join columns' as the primary table.

`-join_columns` *`Column_Names`*

The join columns can be one or more columns that provide a unique key that only matches one row at a time. This usually is a full object name like a net,port or cell name, but can also be a 'key set' like a full_name plus object_class.

`-output_table` *`Table_Name`*

An *optional* output table name where the resultant merged table is stored, default is the primary table.

`-columns` *`Column_Names`*

An *optional* merge column list of columns that specify the columns to merge from the merge table, default is all.

Description

This command allows you to merge two loaded tables with a common set of (key/join) columns and produces a resultant merged table that can be saved under a new table name or by default, will be saved in the original primary table. Both tables have to be loaded in memory, see links for gui_import_utable and gui_open_utable below to read in tables.

Examples

The following example merges two tables (tableA & tableB) joining them via the common key columns (full_name & object_class) and then outputs the result into tableC. It also only merges the column 'power' from tableB with all of tableA, by default all columns in tableB would be merged.

```
shell> # Import tableA:
       gui_import_utable -file tableA.csv
       # Import tableB:
       gui_import_utable -file tableB.csv

       # Merge them into a new tableC
       gui_merge_utable -name tableA -merge tableB -output tableC \\
            -join_columns {full_name object_class} -columns {power}

       # View merged tableC:
       gui_show_utable -name tableC
```

Feedback

See Also

- gui_get_utable
- gui_import_utable
- gui_export_utable
- gui_open_utable
- gui_write_utable

---

gui_mouse_tool

Operate mouse tools of a given view

Syntax

gui_mouse_tool

-window *window* [-start *tool* | -current | -list | -add_point *{x y}* | -drag *{{x1 y1} {x2 y2}}* | -delete_point | -apply | -reset | -cancel | -cycle | -cycle_back]

```
window          String
tool            String
```

Arguments

`-window window`

window specifies the window of which this command should operate the mouse tool.

`-start tool`

Starts the given mouse tool.

`-current`

Return the name of current active tool.

`-list`

Return the names of all available tools.

`-add_point {x y}`

Add an input point

`-drag {{x1 y1} {x2 y2}}`

Perform drag operation between two points.

Feedback

`-delete_point`

Removes last input point

`-apply`

Performs tool action

`-reset`

Clears all pending input or ends the tool if no pending

`-cancel`

Cancels mouse tool unconditionally

`-cycle`

Cycles through choices for a given operation

`-cycle_back`

Cycles back through choices for a given operation

Description

The *gui_mouse_tool* controls the mouse tool of a view that supports mouse actions log and replay capabilities. Not all views need to support all of the functionality provided by the interface of this command. User can use the command to script varios mouse action.

Examples

The following example has the layout view start the zoom in tool and perform a zoom to particular area.

```
> gui_mouse_tool -window Layout.1 -start ZOOM_IN_TOOL
> gui_mouse_tool -window Layout.1 -add_point {27.461 433.400}
> gui_mouse_tool -window Layout.1 -add_point {51.261 393.123}
> gui_mouse_tool -window Layout.1 -cancel
```

---

gui_open_error_data

Opens an error data file in the error browser.

Syntax

```
gui_open_error_data
[-file_name file_name]
[error_data]
```

Data Types

```
file_name        string
error_data       list
```

Feedback

Arguments

`-file_name` *file_name*

Specifies the name of the exported error file to open in the error browser. The name of the error data file to be opened and shown in the error browser. If other error data was already loaded into the error browser, the specified data is added to the existing data. This option is mutually exclusive with the *-name* and *error_data* options.

error_data

Specifies the data name or list of data names to open in the error browser. If the error data is already open, then a collection of the error data or the error data name can be used to show it in the error browser.

Description

The *gui_open_error_data* command displays the given error data in the GUI error browser. If the error browser is not already visible, it is first made visible.

If you specify a file name with the *-file_name* option, the command first opens an error data file with the given file name.

Note that if the command first opens an error data, it increments the open_count for an error data. Incrementing the open_count needs to be balanced by the same number of decrementing the open_count before the error data is closed and removed from memory.

The *gui_open_error_data* command opens an error data only when necessary. It does not increment the open_count if the error data is already open and in memory. If the *gui_open_error_data* command increments the open_count, then a subsequent call to *gui_close_error_data* for the same error data decrements the open_count. Otherwise, a subsequent call to *gui_close_error_data* simply removes the error_data from the error browser without decrementing the open_count.

Examples

The following example opens the physical DRC error data file that is named "my_design_dppinassgn.err" and shows it in the error browser.

```
prompt> gui_open_error_data -file_name my_design_dppinassgn.err]
```

The following example first opens the physical DRC error data file that is named "my_design_dppinassgn.err", then shows it in the error browser.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> gui_open_error_data $data
```

Feedback

See Also

- close_drc_error_data
- collections
- create_drc_error_data
- get_drc_error_data
- gui_close_error_data
- gui_get_error_data
- open_drc_error_data
- remove_drc_error_data

---

gui_open_utable

Open the given UserTable file.

Syntax

string *gui_open_utable*

```
-file         File_Name
[-name        Table_Name]
[-read_only]

File_Name     String
Table_Name    String
```

Arguments

`-file File_Name`

The name of the file to open and copy the UserTable into memory.

`-name Table_Name`

The name of the user table located in the file. By default the name of the file minus the extension is used.

`-read_only`

This flag stops the table from being copied into memory and instead the table contents are streamed which can save a lot of system memory if the table is huge.

Feedback

Description

This command opens the given UserTable file and copies the table contents into memory unless the **-read_only** flag is given which causes the contents to be streamed in as needed. This can save system memory if the table is huge.

Examples

The following example opens the UserTable file 'my_table.utable' and copies the
table 'my_table' into memory.

```
shell> gui_open_utable -file my_table.utable
```

The following example opens the given UserTable file but instead of copying it
into memory it connects in read-only mode, steaming the contents as needed.

```
shell> gui_open_utable -file my_table.utable -read_only
```

See Also

- gui_append_utable
- gui_change_selection_utable
- gui_close_utable
- gui_create_utable
- gui_export_utable
- gui_import_utable
- gui_show_utable
- gui_write_utable

---

gui_overlay_layout

Add/Remove/Adjust a design overlay to a layout window

Syntax

gui_overlay_layout

```
-window windowID
-design design
-add
```

Feedback

```
-remove
-brightness value

windowID        String
design          String
add             Boolean flag
remove          Boolean flag
brightness      Integer
```

Arguments

`-window windowID`

windowID specifies the layout window to add the overlay

`-design design`

design specifies the design to overlay

`-add`

add the design as an overlay

`-remove`

remove the design as an overlay

`-brightness value`

adjust the brightness of the overlay design

Description

Use this command to add, remove, or adjust an overlay on a layout view. The design to overlay must already be an opened design. You cannot overlay the same design twice nor can you overlay a design on itself. The -add and -remove flags are mutually exclusive. The brightness of an overlay can be varied from 0 (not visible at all) and 100 (fully bright). The -remove flag is mutually exclusive with the -brightness flags.

Examples

```
shell> gui_overlay_layout -window Layout.1 -design fill_view -add
shell> gui_overlay_layout -window Layout.1 -design fill_view -brightness
 Off
shell> gui_overlay_layout -window Layout.1 -design fill_view -brightness
 33%
shell> gui_overlay_layout -window Layout.1 -design fill_view -remove
```

---

gui_place_charts_plots

Place plots in charts view

Feedback

Syntax

```
status gui_place_charts_plots
-view view_name
[ -vertical ]
[ -horizontal ]
[ -rows int ]
[ -columns int ]
[ plots ]
```

Data Types

```
view_name     string
plot_id_list  list
```

Arguments

`-view view_name`

Charts View to place plots in.

`-vertical`

Stack plots vertically.

`-horizontal`

Stack plots horizontally.

`-rows int`

Stack plots in grid using the specified number of rows.

`-columns int`

Stack plots in grid using the specified number of columns.

plots

List of plots to place. If not specified then all plots are placed

Description

Place plots in a view.

Examples

The following example stacks all plots horizontally.

```
shell> gui_place_charts_plots -view $view -horizontal
```

See Also

- gui_create_charts_plot

Feedback

gui_query_objects

Get the value of a query text for a collection of object.

Syntax

string *gui_query_objects*

object_collection

Data Types

`string object_collection`

Arguments

object_collection

Specifies the collection of objects to use to get the query text.

Description

The *object_collection* command returns and displays the query text. Same text will be shown in Query toolbar for a particular object query.

A *object_collection* is collection of objects for which the query text will be generated. The collection of objects might be returned as the result of another command such as the *get_cells* command.

For information about collections, see the *collections* man page.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example displays query text for the all plan groups in the design.

```
prompt> gui_query_objects [get_plan_groups]
*********************************
Object summary:
plan_group : 5
*********************************

plan_group : RES_I109

plan_group : RES_I110

plan_group : RES_I111
```

```
plan_group : RES_I112

plan_group : RES_I113
```

The following example displays a query text for cell named RES_I112/nr02d0_B3_1.

```
prompt> gui_query_objects [get_cells RES_I112/nr02d0_B3_1]
*********************************
cell : RES_I112/nr02d0_B3_1
*********************************
allowable_orientation                  N FN FS S FW W E FE
area                                   25.600000
aspect_ratio                           2.500000
bbox                                   {305.600 155.200} {308.800 163.200}
boundary                               {305.600 155.200} {308.800 155.200}
 {308.800 163.200} {305.600 163.200} {305.600 155.200}
cell_id                                3
design                                 test
eco_status                             eco_reset
fp_area                                25.600000
fp_aspect_ratio                        2.500000
fp_height                              8.000000
fp_number_of_hard_macro                0
fp_number_of_io_cell                   0
fp_number_of_macro                     0
fp_number_of_standard_cell             0
fp_width                               3.200000
full_name                              RES_I112/nr02d0_B3_1
height                                 8.000000
is_black_box                           false
is_fixed                               false
is_hard_macro                          false
is_hierarchical                        false
is_io                                  false
is_jtag                                false
is_logical_black_box                   false
is_macro_shape_fixed                   false
is_mim                                 false
is_mim_master_instance                 false
is_physical                            true
is_physical_black_box                  false
is_physical_only                       false
is_placed                              true
is_preserved                           false
is_soft_fixed                          false
is_soft_macro                          false
is_spare_cell                          false
mask_layout_type                       std
movebound                              RES_I112
name                                   nr02d0_B3_1
number_of_pins                         3
object_class                           cell
object_id                              9782
```

Feedback

```
orientation                              N
origin                                   305.600 155.200
outer_keepout_margin                     no_outer_keepout_margin
ref_lib_name
  /remote/dtdata1/testdata/designs/syn/ggui/read_cell_demo/mw/cb25_typ
ref_name                                 nr02d0
ref_view_name                            FRAM
sm_estimation_mode                       unestimated
width                                    3.200000
within_ilm                               false
```

See Also

- collections
- get_selection
- query_objects

---

gui_read_user_map

Reads map mode data of file and display in layout, using user map mode(userMap). The command will show a dialog for user to complete the option values. If user executes the command with options, the option values will sync up to the dialog.

Syntax

string *gui_read_user_map*

```
[-file inputFile]

[-title mapTitle]

[-replace Replace current map if it exists.]

[-preview Preview data in dialogue.]
```

Data Types

```
inputFile     string
mapTitle      string
```

Arguments

`-file` *inputFile*

Specifies the name of the input file. It can be a single file name(generated in previous working directory) or have an absolute path.

For displaying the output data as a user-map, including the meta data is recommanded.

`-title` *`mapTitle`*

Specifies the title of the user map to display.

`-replace`

To replace the current user map.

`-preview`

To preview data in dialogue.

Description

Read map mode data from csv file. User map is a general feature and will not have all of the features of the map that was written only the basic data presentation and bucket settings.

Here's an example of the input file of cell density map with Meta data and header:

```
#META_DATA
#type,name,property,value
#table,,tag,gui_write_user_map
#table,,app_name,icc2
#table,,version,"V1.0"
#table,,report_date,"2029-Jul-29 02:55:51"
#table,,title,"Cell Density"
#table,,bins,10
#table,,min_threshold,0.100000
#table,,max_threshold,1.100000
#table,,color_scale,TemperatureScale
#column,Coordinates,type,poly_rect
#column,value,type,double
#column,Comment,type,string
#END_META_DATA

Coordinates,Value,Comment
"{12.600 12.600} {25.200 0.000}",0.161270,0.16127
"{25.200 12.600} {37.800 0.000}",0.551746,0.55175
"{37.800 12.600} {50.400 0.000}",0.643175,0.64317
"{50.400 12.600} {63.000 0.000}",0.613333,0.61333
"{63.000 12.600} {75.600 0.000}",0.579048,0.57905
```

Example of Global Route Congestion map without Metadata and hear:

```
"{0.000 0.000} {2.520 0.000}",-23,-23/23
"{0.000 2.520} {0.000 0.000}",-26,-26/26
"{2.520 0.000} {5.040 0.000}",-26,-26/26
"{2.520 2.520} {2.520 0.000}",-26,-26/26
"{5.040 0.000} {7.560 0.000}",-24,-24/25
"{5.040 2.520} {5.040 0.000}",-25,-25/26
"{7.560 0.000} {10.080 0.000}",-26,-26/26
```

Feedback

Examples

```
While the layout view is open:

prompt>gui_read_user_map -file cellden.csv
```

See Also

- gui_write_user_map
- gui_remove_user_map

---

gui_remove_all_annotations

Removes specified group of annotations or all annotations from the specified layout window or from all windows.

Syntax

```
status gui_remove_all_annotations
[-window window_name]
[-group group_name]
```

Data Types

```
window_name      string
group_name      string
```

Arguments

`-window window_name`

Specifies the instance name of the layout window.

`-group group_name`

Specifies the annotation group name.

Description

This command removes specified group of annotations from the specified layout window. If group name is omitted, it will remove all annotations groups. If window name is omitted, it will remove annotations from all windows. The command returns 1 if it is successful.

Examples

```
prompt> gui_add_annotation -window Layout.1 -group Test1 \\
   -type rect -text Test1 -color green -width 2 \\
   -pattern Dense5Pattern {{200 200} {777 777}}
1
prompt> gui_remove_all_annotations -window Layout.1
1
```

Feedback

See Also

- gui_add_annotation
- gui_remove_annotations

---

gui_remove_all_rulers

Removes rulers from the specified layout window or from all windows.

Syntax

```
status gui_remove_all_rulers
[-window window_name]
```

Data Types

```
window_name      string
```

Arguments

`-window window_name`

Specifies the instance name of the layout window.

Description

This command removes rulers form specified layout window, or from all windows if the *-window* option is omitted, and returns 1 if it is successful.

Examples

```
prompt> gui_remove_all_rulers -window Layout.1
1
```

See Also

- gui_remove_ruler

---

gui_remove_annotations

Removes specified group of annotations from the specified layout window.

Syntax

```
status gui_remove_annotations
[-window window_name]
[-group group_name]
[anno]
```

Feedback

Data Types

```
window_name      string
group_name      string
anno      collection
```

Arguments

`-window window_name`

Specifies the instance name of the layout window.

`-group group_name`

Specifies the annotation group name.

`anno`

Specifies a collection of annotations to remove. Get a collection of annotations with the gui_get_annotations command. This option cannot be specified with any other option.

Description

This command removes specified group of annotations from the specified layout window. If group name is omitted, global group name is used. If window name is omitted, global window name is used. The command returns 1 if it is successful.

Examples

```
prompt> gui_add_annotation -window Layout.1 -group Test1 \\
   -type rect -text Test1 -color green -width 2 \\
   -pattern Dense5Pattern {{200 200} {777 777}}
1
prompt> gui_remove_annotations -window Layout.1
1
prompt> gui_remove_annotations [gui_get_annotations -filter
 client_data==MyData]
1
```

See Also

- gui_add_annotation
- gui_remove_all_annotations
- gui_get_annotations

---

gui_remove_category_rules

Removes all category rules except built-in rules by default.

Feedback

Syntax

string *gui_remove_category_rules*

```
[-all | -names rule_names_list]
```

Data Types

rule_names_list	list

Arguments

`-all`

Removes all category rules, including built-in rules.

This option and the *-names* option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

`-names` *rule_names_list*

Specifies the names of the category rules to be removed.

This option and the *-all* option are mutually exclusive. If you do not specify either of these options, the command removes all category rules except the built-in rules.

Description

This command removes category rules that have already been created by the *gui_create_category_rule* command during the current run of the tool.

When you use this command without specifying any options, it removes all category rules except built-in rules. Specify the *-all* option to remove all category rules, including the built-in rules. Specify the *-names* option to remove individual rules by name.

The command returns an empty string as its result.

Examples

The following example removes all category rules, including the built-in rules:

```
prompt> gui_remove_category_rules -all
```

The following example removes all category rules except the built-in rules:

```
prompt> gui_remove_category_rules
```

The following example removes the rules named Rule1 and Rule2:

```
prompt> gui_remove_category_rules -names {Rule1 Rule2}
```

Feedback

The following example removes non-built-in rules at the top of a user-defined category rule creation script, which is being developed interactively. You can repeatedly change and refine this script by using a text editor, and you can source the script multiple times in a single run of the tool.

```
        # first remove all existing user-defined non-built-in rules
gui_remove_category_rules
# now create the user-defined category rules
gui_create_category_rule ...
gui_create_category_rule ...
and so forth
```

See Also

- gui_create_category_rule
- gui_list_category_rules

---

gui_remove_cell_block_marks

Removes the block mark string values from one or more cells.

Syntax

status *gui_remove_cell_block_marks*

`-all | cells`

Data Types

cells list

Arguments

`-all`

Removes all block marks on all cells that are marked.

This option and the *cells* option are mutually exclusive, and one of them must be specified.

cells

Removes block marks from one or more cells specified in a list of cell name patterns or cell collections.

This option and the *-all* option are mutually exclusive, and one of them must be specified.

Feedback

Description

This command removes the block mark string values from one or more hierarchical or leaf-level cell instances. If the *-all* option is specified, then all the block marks are removed from all the cell instances that are marked. If the *cells* option is specified, then block marks are removed only from the specified cell instances.

Examples

The following example sets the block mark for a hierarchical cell instance identified by a cell name, and then removes the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
prompt> gui_remove_cell_block_marks I_TOP/I_ALU
```

The following example sets the block mark for a hierarchical cell instance identified by a nested run of the *get_cells* command, and then removes the block mark from that cell instance:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU]  ALU
prompt> gui_remove_cell_block_marks [get_cells I_TOP/I_ALU]
```

The following example removes all the block marks from all the cell instances that are marked:

```
prompt> gui_remove_cell_block_marks -all
```

See Also

- gui_set_cell_block_marks
- gui_get_cell_block_marks
- gui_list_cell_block_marks

---

gui_remove_charts_annotation

Remove annotation object from view or plot

Syntax

```
status gui_remove_charts_annotation
{ -view view_name | -plot chart_id }
{ -id string | -all }
```

Data Types

```
view_name  string
chart_id   string
```

Feedback

Arguments

`-view view_name`

View to remove annotations from.

`-plot chart_id`

Plot to remove annotations from.

`-id string`

Id of annotation to remove.

`-all`

Remove all remove annotations.

Description

Removes one or all annotations from a chart or a view.

Examples

The following example removes all annotations from a view.

```
shell> gui_remove_charts_annotation -view $view -all
```

See Also

- gui_create_charts_arrow_annotation
- gui_create_charts_ellipse_annotation
- gui_create_charts_point_annotation
- gui_create_charts_polygon_annotation
- gui_create_charts_polyline_annotation
- gui_create_charts_rectangle_annotation
- gui_create_charts_text_annotation

---

gui_remove_charts_model

Remove model from charts

Syntax

```
status gui_remove_charts_model
-model model_id
```

Feedback

Data Types

model_id int

Arguments

-model *model_id*

Charts model to remove.

Description

Remove model data from charts.

Note: It is recommended that existing plots using the model should also be removed as they may lose access to some or all of the original data.

Examples

The following example removes a model.

```
shell> gui_remove_charts_model -model $model
```

See Also

- gui_create_charts_model

---

gui_remove_charts_plot

Remove one or all plots from a charts view

Syntax

```
status gui_remove_charts_plot
-view view_name
{ -plot chart_id | -all }
```

Data Types

view_name string
chart_id string

Arguments

-view *view_name*

View to remove chart from.

-plot *chart_id*

Chart to remove.

Feedback

`-all`

Remove all changes.

Description

Removes one or all charts from a view.

Examples

The following example removes all charts from a view.

```
shell> gui_remove_charts_plot -view $view -all
```

---

gui_remove_pref_key

Remove a preference key.

Syntax

string *gui_remove_pref_key* -key *Key*

`[-category Category]`

```
Key             String
Category        String
```

Arguments

`-key Key`

Key specifies the key which will be used together with the specified category to uniquely identify the key to be removed.

`-category Category`

Category specifies the category that owns the specified key. If not specified, a default category will be used.

Description

This command removes the preference key specified by the key name.

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value

Feedback

- gui_get_pref_value_type
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_remove_ruler

Removes the specified rulers.

Syntax

```
status gui_remove_ruler
[-window window_name]
-point point | -all
```

Data Types

```
window_name      string
point            point
```

Arguments

`-window window_name`

Specifies the instance name of the layout window.

`-point point`

Specifies the coordinates of the point near the ruler to be removed, in following format:

```
{x y}
```

`-all`

Removes all rulers.

Description

This command removes rulers from the specified layout window, or from all windows if the *-window* option is omitted, and returns 1 if it is successful.

Examples

```
prompt> gui_remove_ruler -window Layout.1 -point {400 400}
1
```

Feedback

```
prompt> gui_remove_ruler -window Layout.1 -all
1
```

See Also

- gui_remove_all_rulers

---

gui_remove_user_map

Remove current/all user map mode.

Syntax

string *gui_write_user_map*

```
[-all]
```

Arguments

`-all`

Remove all user map mode.

Description

Remove current/all user map mode.

Examples

```
Remove current user map in layout:

prompt>gui_remove_user_map
```

See Also

- gui_read_user_map
- gui_write_user_map

---

gui_remove_vm

Remove Visual Mode

Syntax

string *gui_remove_vm* -name *identifier*

```
string identfier
```

Feedback

Arguments

`-name identifier`

Specifies the name of the visual mode to be removed.

Description

Removes the specified visual mode and all the associated buckets.

Examples

To remove visual mode named "mycoloring1", use the following command:

shell> *gui_remove_vm -name mycoloring1*

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm
- gui_get_vmbucket
- gui_list_vm
- gui_remove_vmbucket
- gui_set_vmbucket
- gui_update_vm

---

gui_remove_vmbucket

Removes the specified bucket or all buckets from the specified visual mode

Syntax

string *gui_remove_vmbucket*

-vmname *mode_identifier* [-name *bucket_identifier*] [-all]

Data Types

mode_identifier	string
bucket_identifier	string

Feedback

Arguments

`-vmname` *mode_identifier*

Specifies the visual mode of which the bucket is a member. The visual mode must already exist.

`-name` *bucket_identifier*

Specifies the name of the bucket to be removed from the specified visual mode.

`-all`

Removes all buckets associated with the specified visual mode.

Description

Removes the specified bucket or all buckets from the specified visual mode.

Examples

To remove the visual mode bucket named red from the visual mode named vm1, enter the following command:

```
prompt> gui_remove_vmbucket -vmname vm1 -name red
```

To remove all the visual mode buckets from the visual mode named vm1, enter the following command:

```
prompt> gui_remove_vmbucket -vmname vm1 -all
```

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm
- gui_get_vmbucket
- gui_list_vm
- gui_remove_vm
- gui_set_vmbucket
- gui_update_vm

---

gui_report_errors

Output a textual report of errors.

Feedback

Syntax

string *gui_report_errors*

[-errors *Errors*] [-file *FileName*] [-include_hidden]

```
Errors          String
FileName        String
```

Arguments

`-errors Errors`

The *-errors* specifies the handle for a collection of errors to report on. This argument is optional. If omitted, a report is output for the errors currently loaded in the Error Browser error list.

`[-file FileName]`

FileName specifies the name of the file to write the report to. This argument is optional. If omitted, the report is output to stdout.

`[-include_hidden]`

-include_hidden specifies that errors that are currently hidden from the errors list due to filters or fixed errors being hidden should be included in the report. This option is ignored if errors are given with the *-errors* option.

Description

This command outputs a tabular textual representation of errors. If the -errors argument is omitted and an output file is specified, performs the same operation as the Error Browser option menu command to save to file.

Examples

The following example saves a textual report of current errors to a file named myErrors.txt:

gui_report_errors -file "myErrors.txt"

See Also

- gui_error_browser
- gui_set_current_errors

---

gui_report_hotkeys

Report on the current hotkey bindings

Feedback

Syntax

string *gui_report_hotkeys*

```
[-window_type WindowTypeName]

WindowTypeName  String
```

Arguments

`-window_type` *WindowTypeName*

WindowTypeName specifies the type of top level window that the hotkey should apply to. (A top level window is a window with a menu bar.) If it is not specified then the applications default window type will be used. The read-only variable gui_default_window_type(3) specifies the default window type for the application.

Description

This command prints a report on the current key bindings. The list of bindings printed in the report are sorted by hotkey.

Examples

The following example creates a menu item and add an additional hotkey to it.

```
gui> gui_report_hotkeys

*************************************
Report : hotkeys
Window : LayoutWindow
Version: V-2004.06-GALILEO-BETA1-1
Date   : Tue Aug 31 10:37:17 2004
*************************************

Hot Key    Type  Function
-----------------------------------------------------
Ctrl+O     Menu  File->Open Design...
Ctrl+S     Menu  File->Save Design
Ctrl+R     Menu  Edit->Properties
M          Menu  Edit->Move...
C          Menu  Edit->Copy...
Ctrl+F     Menu  View->Zoom->Zoom Full
F          Menu  View->Zoom->Zoom Full
P          Tcl   query_objects [get_selection]
 ...
```

See Also

- gui_set_hotkey
- gui_create_menu

Feedback

- gui_delete_menu
- gui_create_toolbar
- gui_delete_toolbar
- gui_create_toolbar_item
- gui_delete_toolbar_item
- gui_default_window_type
- gui_custom_setup_files

---

gui_report_map

Report information for a specified map mode.

Syntax

string *gui_report_map*

```
-name identifier
-report_type outputType
```

Data Types

```
identifier      string

outputType      string (Values: histogram, count)
```

Arguments

`-name` *identifier*

Specifies the name of the map mode to be queried.

`-report_type outputType`

Specifies the mode of report. Value of *outputType* can be *histogram* or *count*.

Description

The *gui_report_map* command queries the option values of an existing map mode name and a report mode. Histogram mode will print '###' as a graph bar for each bucket. Count mode will generate only count numbers of blocks in each bucket.

Feedback

Examples

To generate a histogram data for the map mode named "cellDensityMap". Use the following command:

```
prompt>gui_report_map -name cellDensityMap -report_type histogram
**************************************
Report : map
Design : ORCA
Mode   : histogram
Version: P-2019.03-DEV
Date   : Tue Nov  6 22:01:35 2018
**************************************
[1.00 - 1.10)   ( 0 )
[0.90 - 1.00)   ( 0 )
[0.80 - 0.90)   ( 0 )
[0.70 - 0.80)   ( 0 )
[0.60 - 0.70)   ( 0 )
[0.50 - 0.60)   ( 0 )
[0.40 - 0.50) ############### ( 269 )
[0.30 - 0.40)
 ############################################################## ( 1200 )
[0.20 - 0.30) ##################### ( 379 )
[0.00 - 0.20) ###### ( 104 )
```

See Also

- gui_get_map
- gui_get_mapbucket

---

gui_report_performance

Generates a performance report.

Syntax

```
status gui_report_performance
[-type report_type]
[-file file_name]
[-compress compress_method]
```

Data Types

```
report_type       string
file_name          string
compress_method   string
```

Feedback

Arguments

`-type` *report_type*

Specifies the report type. If report type is omitted or the value of this option is simple, output a simple version report. If the value of this option is detail, output a detail version report which includes additional information of x11 performance, region searching, attribute querying and render time.

The supported values for this option are

simple
detail

`-file` *file_name*

Outputs the report as a file and specifies the file name. If file name is omitted, output the report to console.

`-compress` *compress_method*

Specifies the compress method. The supported values for this option are

none
gzip

Description

This command report the performance data of machine state, design information, GUI configurations and current GUI views state.

Examples

```
Generate a simple version report and output to console.

prompt> gui_report_performance

Generate a detail version report and output to a file.

prompt> gui_report_performance -type detail -file file_name

Generate a detail version report and output to a compressed file

prompt> gui_report_performance -type detail -file file_name -compress
 gzip
```

---

gui_report_proc_arg_type_names

Report on the available type_name attribute values.

Feedback

Syntax

string *gui_report_proc_arg_type_names*

Description

This command prints a report on the available values for the type_name attribute for the *-define_args* argument to the *define_proc_attributes* command.

When a tcl proc has been defined using the *define_proc_attribute* command, a GUI form for the command will be auto-generated and shown from the application command search feature or using the command *gui_show_command_form*. The kinds of proc argument input fields used in the generated GUI form depend on the *type_name* attribute values provided for command options. The *type_name* attribute value for an option is given as a part of the *-define_args* argument. The *-define_args* option accepts a list of the following values, in this order:

```
* option name
* option decription
* option value type description
* data_type
* list of option attribute name-value pairs
```

Here is the usage for the *type_name* attribute within argument value for the *define_proc_attributes* command option *-define_args*:

```
define_proc_attributes <procedure name> \\
  -define_args { \\
    {<option name> <option description> <value type description>
 <data_type> \\
      {[required | optional] {type_name <type_name_value>}} \\
  } \\
}
```

where <type_name_value> can be substituted with any of the type names reported by *gui_report_proc_arg_type_names*.

For the *type_name* attribute value to have effect on the input field used in a GUI form, the <data_type> value must be *string*, for a single value, or *list* for multiple values. The value *list* is not supported for all *type_name* values as shown in the report's *data_type* column.

The following is an example of a *define_proc_attribute* command invocation with options accepting a collection or names of nets:

```
define_proc_attributes myProc \\
  -define_args { \\
    {-net "select a net" "net name or a collection of a net" string \\
          {optional {type_name net}}} \\
    {-nets "select nets" "net names or a collection of nets" list   \\
          {optional {type_name net}} \\
```

```
  } \\
}
```

In the above example, the *-net* option has *data_type* value *string* and *type_name* value *net*. This option accepts a single net as input. The *-nets* option has *data_type* value *list* and *type_name* value *net*. This option accepts multiple nets as input.

The report produced by this command lists the available *type_name* values. When one of these supported *type_name* values is used, the GUI form for the proc will use an input field that is appropriate for user input of that kind of value. For example, for a *net* type input, as shown in the example above, the form will use an object selector field which is configured to accept net input.

A proc argument with any combination of unspecified or unsupported *data_type* and *type_name* values is represented with a simple text input field in the generated GUI form.

Some *type_name* values use input editors that allow further configuration. For example, an editor for a file name can be configured to allow input of only existing files. Or an editor for a layer name can be configured to allow input of only certain types of layers.

Available configuration options are shown in the options column, then described in more detail at the end of the report output in a separate table keyed by associate type_name values. When providing optional configuration, list the configuration name-value pairs after the type_name value as a part of the the type_name attribute value. The following is an example of a "file" type_name attribute with multiple optional configuration settings:

```
define_proc_attributes myProc \\
  -define_args { \\
    {-file_option "Select an existing file name" "file name"  string \\
      {required
        {type_name {file {subtype exist} \\
                         {operation "Save As"} \\
                         {filter {"Text files (*.txt)" "All files (*)"}}
 \\
                   } \\
        } \\
      } \\
    } \\
  }
```

Examples

The following example outputs a report of available type_name attribute values.

```
gui> gui_report_proc_arg_type_names
****************************************************************************
*****
Report : Supported Values for "type_name" Option Attribute
Version: L-2016.03-SP5-BETA
Date   : Fri Sep 16 08:09:34 2016
```

Feedback

```
*****************************************************************************
*****

type_name                  data_type       description     options
-----------------------------------------------------------------------------
-----
PathGroup                  string, list    object type
bbox                       string, list    region
block                      string, list    object type
block_via_def              string, list    via_def
bool                       string          bool
boolean                    string          boolean
bound                      string, list    object type
bundle                     string, list    object type
cell                       string, list    object type
clock                      string, list    object type
collection                 string          collection
corner                     string, list    object type
cut_pattern                string          cut_pattern
directory                  string          directory name  context,
 filter,
                                                           operation,
 subtype
 ...


Configuration Options:

type_name   option      description
-----------------------------------------------------------------------
layer       subtype     Layer type -- valid values: routing, contact,
                        cut, poly, poly_cont, user, system.  Multiple
                        values may be given in a tcl list.

directory,  context     A valid directory name for current directory
file                    context for the file dialog
            filter      File or directory name filter strings to
                        set for the file dialog.  Valid value is
                        a tcl list of filter strings, e.g. {"Text
                        files (*.txt)" "All files (*)"}
            operation   Operation name for file dialog titlebar
                        and button, e.g. "Save As"
            subtype     File type: valid values are: "existing"
                        (existing file only), "any" (any file)
```

See Also

- define_proc_attributes
- gui_show_command_form

Feedback

gui_report_task

task.

Syntax

string *gui_report_task*

```
-task | -item_root          TaskName | ItemRootName
[-file]                     FileName

TaskName          String
ItemRootName      String
FileName          String
```

Arguments

`-task TaskName`

TaskName specifies the name of a task on which to report. This option is mutually exclusive with the -item_root option.

`-item_root ItemRootName`

ItemRootName specifies the name of a task item root name on which to report. This option is mutually exclusive with the -task option.

`-file FileName`

FileName optionally specfies the output file into which the report is written.

Description

This command outputs a textual report of the given task to the xterm console. If -file is given, then the report is also output to the given file. The report will have the following approximate format. The strings bracketed with the angle brackets will be replaced with the actual attribute value.

Attributes are omitted if they are not defined. Item are shown one to a line with the hierarchy structure indicated by indentations and bars. The leaf-leveltask item names are followed by the associated task page name:

```
Task: <task name>
 Default: <true | false>
 Items: <items>
```

See Also

- gui_create_task
- gui_get_task_list

Feedback

gui_reset_eco_context

Resets the context for manual ECO in the GUI for a given type if specified or for all the types.

Syntax

```
status gui_reset_eco_context
[-type ype of context]
```

Data Types

```
type        string
```

Arguments

type

Resets the context for the given type. Supported types are 'insert_buffer' or 'size_cell'. If no type is specified, it will reset both the types.

Description

The *gui_reset_eco_context* command resets the context of manual ECO in the GUI.

The *gui_reset_eco_context* command can be used to reset the context and *gui_set_eco_context* can be used to set the context in the GUI for manual ECO. *gui_get_eco_context* can be used to get the context set for a given 'type'.

Examples

The following example shows how to reset a given context.

```
prompt> gui_reset_eco_context -type "insert_buffer"
1
prompt> gui_reset_eco_context
1
```

See Also

- gui_set_eco_context
- gui_get_eco_context

gui_schematic_add_logic

Adds logic to a schematic

Syntax

string *gui_schematic_add_logic* [-window <*win*>]

Feedback

[-new] [-schematic <*schem*>] *objs*

```
string <win>
string <schem>
string objs
```

Arguments

`-window <win>`

Top level window name to place new schematic (default: current window). This option is ignored unless -new is specified.

`-new`

Create new schematic.

`-schematic <schem>`

Schematic view to update (default: most recently active schematic).

objs

Objects to add to the schematic.

Description

This command adds logic into a Path Schematic view. The command either creates a new view containing the specified objects or updates an existing view. The command always returns the name of the schematic that was updated or created.

Examples

The following example creates a new path schematic containing all the cells in the current design:

```
gui_schematic_add_logic -new [get_cells *]
```

See Also

- gui_get_current_window
- gui_schematic_remove_logic

---

gui_schematic_remove_logic

Removes logic from a schematic

Syntax

string *gui_schematic_remove_logic* [-schematic <*schem*>] *objs*

Feedback

```
string <schem>
string objs
```

Arguments

`-schematic <schem>`

Schematic view to update (default: most recently active schematic).

objs

Objects to remove from the schematic.

Description

This command removes logic into a Path Schematic view. The command returns the name of the schematic that was updated

Examples

The following example removes the cell "U1" from the most recently active schematic view:

```
gui_schematic_remove_logic [get_cells U1]
```

See Also

- gui_get_current_window
- gui_schematic_add_logic

---

gui_scroll

Scroll a window's viewport.

Syntax

gui_scroll

-window *window* [-selection | -clct *clct* | [-habs *absX* | -hrel *relX*] [-vabs *absY* | -vrel *relY*]]

```
window String
absX  Float
relX  Float
absY  Float
relY  Float
```

Arguments

`-window window`

window specifies the window to be scrolled.

Feedback

`-selection`

Determine the bounding box for all selected objects present in the view, and scroll such that the center of that box is visible.

`-clct` *`clct`*

Determine the bounding box for the collection objects present in the view, and scroll such that the center of that box is visible.

`-habs` *`absX`*

Scroll such that the specified X model position is centered in the viewport.

`-hrel` *`relX`*

Scroll horizontally by the specified number of pages. A page is the width of the viewport. Negative values scroll left and positive values scroll right.

`-vabs` *`absY`*

Scroll such that the specified Y model position is centered in the viewport.

`-vrel` *`relY`*

Scroll vertically by the specified number of pages. A page is the height of the viewport. Negative values scroll up and positive values scroll down.

Description

The *gui_scroll* command adjusts the position of views that support zooming. The viewport can be adjusted in the X and/or Y direction. Absolute values are expressed in model coordinates. Relative values allow paging. For example, *-hrel 1.0* means page to the right by the width of the viewport.

Examples

Scroll such that model coordinates (287.0, 1422.0) are centered in the viewport.

```
gui_scroll -window Layout.1 -habs 287.0 -vabs 1422.0
```

Scroll to the right by one viewport width.

```
gui_scroll -window Layout.1 -hrel 1.0
```

See Also

- gui_zoom

---

gui_select_by_name

Select or highlight objects by name

Feedback

Syntax

```
status gui_select_by_name
[ { -select | -highlight } ]
-object_type string
[ -pattern string ]
[ -filter string ]
[ { -replace | -add | -remove } ]
[ -regexp ]
[ -nocase ]
[ -exact ]
[ -all ]
```

Arguments

`-select`

Specifies that the matching objects are to be selected.

Note: only one of *-select* or *-highlight* options should be specified. If neither is specified, *-select* is assumed.

`-highlight`

Specifies that the matching objects are to be highlighted.

Note: only one of *-select* or *-highlight* options should be specified. If neither is specified, *-select* is assumed.

`-object_type string`

Specifies the type of objects to be selected.

The available types are dependant on the individual product. Please see your product's documentation for details.

`-pattern string`

Specifies the pattern to be used for a match.

By default matching is case-sensitive and uses wildcard patterns. If non case-sensitive matching is required use the *-nocase* option. If regular expression matching is required use the *-regexp* option. If no pattern matching is required use the *-exact* option.

`-filter string`

Specifies the expression to use when filtering the results.

`-replace`

Specifies that matching objects should replace the current selection or highlight.

Note: only one of the *-replace*, *-add* or *-remove* options can be specified. If none of these options area specified, then replace is assumed.

Feedback

`-add`

Specifies that matching objects should be added to the current selection or highlight.

Note: only one of the *-replace*, *-add* or *-remove* options can be specified. If none of these options area specified, then replace is assumed.

`-remove`

Specifies that matching objects should be removed from the current selection or highlight.

Note: only one of the *-replace*, *-add* or *-remove* options can be specified. If none of these options area specified, then replace is assumed.

`-regexp`

Specifies the pattern matching should use regular expressions.

`-nocase`

Specifies the pattern matching should be case-insensitive.

`-exact`

Specifies that no pattern matching should be used so wildcards and regular expressions are ignored.

`-all`

Specifies that physical only objects be include in the search.

Description

This command allows objects of a specified type to be selected or highlighted.

It is used by the Select By Name Toolbar and Select By Name Dialog for correct log and replay of the resultant selection or highlight.

The command returns whether the search was successful or not.

Examples

The following example will add all nets matching the pattern 'A*' to the current selection.

```
shell> gui_select_by_name -select -object_type {Nets} -pattern {A*} -add
```

The following example will only highlight the Port '*Logic0*'.

```
shell> gui_select_by_name -highlight -object_type {Ports} -pattern
 {*Logic0*} -exact
```

Feedback

See Also

- change_selection
- gui_change_highlight

---

gui_select_vmbucket

Select the Contents of a Visual Mode Bucket

Syntax

string *gui_select_vmbucket* -vmname *mode_identifier*

-name *bucket_identifier* [-replace | -add | -remove]

```
string  mode_identifier
string  bucket_identifier
```

Arguments

`-vmname mode_identifier`

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes, use the *gui_list_vm* command.

`-name bucket_identifier`

Specifies the name of the visual mode bucket.

`-replace`

Optional argument that indicates that the contents of the visual mode bucket will replace the selection list.

`-add`

Optional string that indicates that the contents of the visual mode bucket will be added to the selection list.

`-remove`

Optional string that indicates the contents of the visual mode bucket will be removed from the selection list.

Description

The *gui_select_vmbucket* command selects the contents of a visual mode bucket. The visual mode bucket is a member of a specific visual mode. The contents of the visual mode bucket can replace, be added to, or be removed from the selection list.

Feedback

Examples

Select the 2nd bucket of SNAPSHOT visual mode.

shell> *gui_select_vmbucket -vmname SNAPSHOT -name 1 -replace*

See Also

- gui_create_vm
- gui_get_vm
- gui_get_vmbucket
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vm
- gui_set_vmbucket
- gui_update_vm

---

gui_selection_stack

Command to manipulate selection stack

Syntax

```
status gui_selection_stack
{ -push | -pop | -clear | -get n | -remove n | -list }
```

Arguments

`-push`

Push the global selection onto the top of the selection stack

`-pop`

Pop the selection from the top of the selection stack and make it the global selection.

`-clear`

Clear the selection stack.

`-get n`

Get the nth element of the selection stack.

Feedback

`-remove n`

Remove the nth element of the selection stack.

`-list`

List the selection stack.

Description

Allows the user to push the global selection onto a stack to be popped off later.

Note: The selection stack has a maximum depth (default 10) to prevent the stack getting too large. Any selection pushed onto the stack when it is full will remove the oldest (bottommost) entry in the stack.

The 'gui_selection_stack_depth' variable can be set to change the depth.

Examples

The following example pushes the global selection to the stack, clears the global selection, and restores the pushed selection.

```
shell> gui_selection_stack -push
shell> change_selection
shell> gui_selection_stack -pop
```

See Also

- gui_selection_stack
- gui_selection_stack_depth

---

gui_set_active_window

Make the specified window the active window

Syntax

```
status gui_set_active_window
-window window_id
```

Data Types

```
window_id    string
```

Arguments

`-window window_id`

Specifies the window name id.

The matching window of this id will be made the active window.

Feedback

Note: both toplevel windows and child windows of a particular toplevel window have a currently active window. If the window is a toplevel window then the currently active toplevel window is changed, of the window is a child window the currently active child window in the specified window's parent is changed.

Description

This command makes the specified window the active window.

Examples

The following example will make the toplevel window Toplevel.2 active.

```
shell> gui_set_active_window -window Toplevel.2
```

See Also

- gui_close_window
- gui_exist_window
- gui_show_window
- gui_get_window_types
- gui_get_window_ids
- gui_create_window
- gui_get_current_window

---

gui_set_bucket_option

Sets the value for an option on a bucket in the specified visual mode or map mode.

Syntax

```
string gui_set_bucket_option
-map map_name
-bucket bucket_name
-option option_name
-value value | -default
```

Data Types

```
map_name          string
bucket_name       string
option_name       string
value             string
```

Feedback

Arguments

`-map` *`map_name`*

Specifies the name of the visual mode or map mode.

`-bucket` *`bucket_name`*

Specifies the name of the bucket.

`-option` *`option_name`*

Specifies the name of the option to be set.

`-value` *`value`*

Specifies the value for the option.

The *-value* option is mutually exclusive with the *-default* option. You must use one, but not both.

`-default`

Sets the option to its default value.

The *-default* option is mutually exclusive with the *-value* option. You must use one, but not both.

Description

This command sets the value for an option on a bucket in a visual mode or map mode. You must specify the visual mode or map mode name, the bucket name, and the option name. You must set the option to either a specific value or its default value. Use the *-value* option to set a specific value or use the *-default* option to set the option to its default value.

Examples

The following example sets the color of *bucket1* to yellow in the global route congestion map:

```
prompt> gui_set_bucket_option -map AREAPARTITION \\
        -bucket bucket1 -option color -value yellow
```

See Also

- gui_get_bucket_option
- gui_get_bucket_option_list
- gui_get_map_list
- gui_get_map_option

Feedback

- gui_get_map_option_list
- gui_set_map_option

---

gui_set_cell_block_marks

Sets a block mark on one or more cells.

Syntax

status *gui_set_cell_block_marks*

```
cells
block_mark
```

Data Types

```
cells          list
block_mark     string
```

Arguments

cells

Lists one or more cells to mark with a block mark, in a list of cell name patterns or cell collections.

block_mark

Specifies the block mark string value to be set for the listed cells.

Description

This command sets a block mark for one or more hierarchical or leaf-level cell instances. Typically, a short symbolic string name is used as the block mark. Ultimately, a block mark can appear in the GUI as the text label for a timing path category generated from a dynamic category rule. Such block marks should be kept short to avoid using up too much screen space in the GUI.

A common application involves marking certain interesting intellectual property (IP) blocks within a design. These block marks affect the calculation of the following timing path shell attributes, which are defined only when the GUI is running:

- *blocks* - An ordered block level path; includes start, through, and end blocks
- *through_blocks* - Includes through blocks, but not the start and end blocks
- *start_block* - The start block
- *end_block* - The end block

- *start_end_blocks* - A pair consisting of the start block and the end block, in that order
- *start_end_blocks_sorted* - A pair consisting of the start block and the end block, sorted alphabetically

These block marks also affect the calculation of the following cell shell attribute, which is defined only when the GUI is running:

- *block_mark* - String value of a cell instance's (explicit) block mark, or an empty string if there is no explicit block mark

In calculating the timing path shell attributes listed previously, a block is calculated for each timing point of a timing path. If the leaf cell instance on which a timing point resides is explicitly marked with a block mark, by using the *gui_set_cell_block_marks* command, the block for the timing point is the block mark for the leaf cell. If the leaf cell instance on which a timing point resides is not explicitly marked, the block for the timing point is the block mark on the first explicitly marked hierarchical cell located by traversing up the logical hierarchy from that leaf cell instance.

If no direct or indirect hierarchical parent cell is found to be explicitly marked, the block for the timing point has a default implicit string value of "_Top_". For a timing point that is a startpoint or endpoint of a timing path, where the startpoint or endpoint is an external design port, the block has a default implicit string value of "_Port_".

Note: Each timing point of a timing path always has a defined block, which can be an explicit block mark, "_Top_", or "_Port_".

The *through_blocks* attribute value can be an empty set of blocks, which is represented by the string value "_Empty_".

Examples

The following example sets the block mark to ALU for a hierarchical cell instance identified by a cell name:

```
prompt> gui_set_cell_block_marks I_TOP/I_ALU  ALU
```

The following example sets the block mark to ALU for a hierarchical cell instance identified by a nested run of the *get_cells* command:

```
prompt> gui_set_cell_block_marks [get_cells I_TOP/I_ALU]  ALU
```

The following example sets the block mark for a list of cell instances where the first item is identified by a cell name and the second item is identified by a nested run of the *get_cells* command:

```
prompt> gui_set_cell_block_marks [list I_ALU  [get_cells
 I_STACK_TOP/I1_STACK_MEM]]  \\
       my_mark
```

Feedback

See Also

- gui_get_cell_block_marks
- gui_remove_cell_block_marks

---

gui_set_charts_data

Set charts data

Syntax

```
status gui_set_charts_data
{ -global | -model model_id | -view view_name |
-plot plot_name | -annotation string }
[ -column int ]
[ { -header | -row int } ]
-name string
-value string
[ -data string ]
```

Data Types

```
model_id   int
view_name  string
plot_name  string
```

Arguments

`-global`

Set global data.

`-model model_id`

Model to set data in.

`-view view_name`

View to set data in.

`-plot plot_name`

Plot to set data in.

`-annotation string`

Set annotation data

`-column int`

Column to set model data in.

Only used by the *model* option.

Feedback

`-header`

Set model header data.

Only used by the *model* option.

`-row` *int*

Row to set model data in.

Only used by the *model* option.

`-name` *string*

Name of data to set.

The supported names depend on which object the data is being set in:

For model the following names are supported:

value Set value of model horizontal header or row value.

column_type Set specific column type or all model column types. If the column is specified then this is the type for the specified column, if no column is specified this is a list of types for the model's columns. See *gui_create_charts_model column_type* for details of the syntax.

name Set model name.

For view the following names are supported:

fit fit plots in the view.

For plot no names are currently supported:

For global no names are currently supported:

`-value` *string*

The value to set.

`-data` *string*

Extra data value

Description

Set data in model, view, plot or globally.

If model is specified then the *column* and *header* or *row* should be specified.

Examples

The following example sets the name of a model.

```
shell> gui_set_charts_data -model $model -name "Test Model"
```

Feedback

See Also

- gui_get_charts_data

---

gui_set_charts_property

Set charts property

Syntax

```
status gui_set_charts_property
{ -view view_name | -plot plot_name |
-annotation annotation_id }
-name string
-value string
```

Data Types

```
view_name       string
plot_name       string
annotation_id   string
```

Arguments

`-view view_name`

View to set property in.

`-plot plot_name`

Plot to set property in.

`-annotation annotation_id`

Plot annotation to set property in.

`-name string`

Name of property to set.

The supported names depend on which object the data is being set in:

`-value string`

The value to set.

Description

Set property of view, plot or plot annotation.

Examples

The following example sets the line stroke color of a plot to red.

Feedback

```
shell> gui_set_charts_property -plot $plot -name "stroke.color" -value
 red
```

See Also

- gui_get_charts_property

---

gui_set_current_errors

Sets the current error data and the current error group in the Error Browser.

Syntax

string *gui_set_current_errors*

[-data_name *ErrorDataName*] [-group *Group*]

```
ErrorDataName       String
Group         String
```

Arguments

`-data_name ErrorDataName`

ErrorDataName specifies the error data to make current. The error data name may be omitted. If omitted, makes all open error data current.

`-group Group`

Group specifies the group name, either a layer name or error type name, to make current. The group name may be omitted. If omitted, an entire error data file is made current.

Description

Sets the current error data and the current error group in the Error Browser. This is analogous to selecting error data and group in the Error Browser tree view. By default, there is are no current errors. The current errors determine the errors that are displayed in the error list and in the layout view by default.

If the error data option is omitted, selects the design, the parent of all open error data. The group option is valid only when there is a single error data specified, and raises an error if no error data was specified. The group string is interpreted as error type if error grouping has been set to *type*, else the group string is interpreted as layer. If the group option is omitted, all groups in the specified error data are made current.

Feedback

Examples

The following example sets error grouping by error type, and sets the "Open Locator" type errors for LVS error data as the current errors:

gui_set_error_browser_option -grouping type gui_set_current_errors -data_name LVS -group "Open Locator"

The following example sets the design as current, displaying all errors in all open error data in the error list and layout view.

gui_set_current_errors

See Also

- gui_error_browser
- gui_set_selected_errors

---

gui_set_current_task

Set current task to the given task.

Syntax

string *gui_set_current_task* -name *task_name*

-task *task_name*

`string task_name`

Arguments

`-task task_name`

Specifies the name of the task to make current.

Description

The gui_set_current_task command sets the current application task to the given task.

See Also

- gui_create_task
- gui_get_current_task
- gui_get_task_list

Feedback

gui_set_eco_context

Sets context for manual ECO in the GUI

Syntax

```
status gui_set_eco_context
[-type Type of context]
[-lib_cells Set library cell]
[-filter_using_attributes Use the following attributes to filter on
 library cells]
```

Data Types

```
type        string
lib_cells   list
filter_using_attributes   string
```

Arguments

type

Set the type of context. Supported types are 'insert_buffer' or 'size_cell'.

lib_cells

Set the library cells for the given context.

filter_using_attributes

Uses the attributes from this list to filter on the library cells. Supported types are 'dont_touch' and 'dont_use'

Description

The *gui_set_eco_context* command sets the context of manual ECO in the GUI.

When GUI ECO dialogs are up and running the library cell is controlled using these set of commands. The *gui_reset_eco_context* command can be used to reset the context and *gui_reset_eco_context* can be used to get the current context given a 'type'.

Examples

The following example shows how a context can be set for a given type.

```
prompt> gui_set_eco_context -type insert_buffer -lib_cells {INV* BUFF*}
 -filter_using_attributes {dont_touch}
1
prompt> gui_set_eco_context -type size_cell -lib_cells $lib_cell_clct
 -filter_using_attributes {dont_use}
1
```

Feedback

See Also

- gui_get_eco_context
- gui_reset_eco_context

---

gui_set_error_browser_option

Sets options for the Error Browser dialog box.

Syntax

string *gui_set_error_browser_option*

[-grouping *Group*] [-show_mode *ShowMode*] [-view_mode *ViewMode*] [-zoom_factor *ZoomFactor*] [-hide_fixed *DoHide*] [-hide_ignored *DoHide*] [-advance_to_next_unfixed *DoNext*] [-dim *DoDim*] [-show_open_locator_nodes *DoShow*] [-show_tooltip *DoShow*] [-show_command_buttons *DoShow*] [-highlight_objects *DoHighlight*] [-auto_set_object_visible *SetVisible*] [-auto_set_object_visible_type *SetVisibleType*]

```
Group              String
ShowMode           String
ViewMode           String
ZoomFactor         float
DoHide             Boolean
DoNext             Boolean
DoDim              Boolean
DoShow             Boolean
DoHighlight        Boolean
SetVisible         Boolean
SetVisibleType     String
```

Arguments

`-grouping Group`

The *-grouping* option accepts argument values *type* or *layer* and sets whether errors are grouped by types or layers. This is similar to selecting or deselecting *Show by Layer* option in the Error Browser *Options* menu. By default, errors are grouped by type by default.

`-show_mode ShowMode`

The *-show_mode* option accepts argument values *all*, *selected*, or *none* and sets whether all errors, selected errors, are no errors are shown in layout views. This is similar to making a selection from the *Show* field in the Error Browser. The default is that all errors are shown.

Feedback

`-view_mode` *`ViewMode`*

The *-view_mode* option accepts argument values *zoom*, *pan*, or *off* and sets the layout view to zoom to, pan to, or not change to the currently selected errors in the Error Browser. This is similar to making a selection from the *Follow* field in the Error Browser. The default is to zoom to the selected error.

`-zoom_factor` *`ZoomFactor`*

The *-zoom_factor* option accepts float values between *0.1* and *10.0* and sets the zoom factor used when the view_mode is *zoom*. The zoom factor rounds to 1/10 precision. The default value is *1.0*.

`-hide_fixed` *`DoHide`*

The *-hide_fixed* option accepts Boolean values *1* or *true* to mean "hide", and *0* or *false* to mean "show". The option controls whether fixed errors are hidden or shown in the Error Browser and layout views. This is similar to selecting or deselecting *Hide Fixed Errors* option in the Error Browser *Options* menu. By default, fixed errors are shown.

`-hide_ignored` *`DoHide`*

The *-hide_ignored* option accepts Boolean values *1* or *true* to mean "hide", and *0* or *false* to mean "show". The option controls whether ignored errors are hidden or shown in the Error Browser and layout views. This is similar to selecting or deselecting *Hide Ignored Errors* option in the Error Browser *Options* menu. By default, ignored errors are shown.

`-advance_to_next_unfixed` *`DoNext`*

The *-advance_to_next_unfixed* option accepts Boolean values *1* or *true* to mean advance to next unfixed error when an error is fixed and *0* or *false* to advance to next fixed/unfixed error. This is similar to checking or unchecking the *Advance To Next Unfixed When Error is Fixed* option in the Error Browser. By default, unfixed errors are not advanced to.

`-dim` *`DoDim`*

The *-dim* option accepts Boolean values *1* or *true* to mean "dim" and *0* or *false* to mean "do not dim". The option governs whether layout views are dimmed when errors are displayed. This is similar to checking or unchecking the *Dim* option in the Error Browser. By default, layout views are not dimmed.

`-show_open_locator_nodes` *`DoShow`*

The *-show_open_locator_nodes* option accepts Boolean values *1* or *true* to mean "show", and *0* or *false* to mean "hide". This setting controls whether net nodes are highlighted together with the open locator flylines for selected open locator errors. This is similar to selecting or deselecting the *Display Net Nodes*

for Selected Open Locators option in the Error Browser *Options* menu. By default, the tool does not show the net node highlights.

`-show_tooltip` *`DoShow`*

The *-show_tooltip* option accepts Boolean values *1* or *true* to mean "show", and *0* or *false* to mean "hide". This setting controls whether tool tips are shown on the error browser list view and details pane. By default, the tool shows the tool tips.

`-show_command_buttons` *`DoShow`*

The *-show_command_buttons* option accepts Boolean values *1* or *true* to mean "show", and *0* or *false* to mean "hide". This setting controls whether the command buttons at the bottom of the error browser are shown. By default, the tool shows the command buttons.

`-highlight_objects` *`DoHighlight`*

The *-highlight_objects* option accepts Boolean values *1* or *true* to mean "highlight", and *0* or *false* to mean "do not highlight". This setting controls whether design objects associated with the selected error are highlighted in the layout view using design object highlighting. When the error object selection changes, the highlight is automatically removed from the associated design objects. By default, the tool does not highlights associated design objects.

`-auto_set_object_visible` *`SetVisible`*

The *-auto_set_object_visible* option accepts Boolean values *1* or *true* to mean "turn on" and *0* or *false* to mean "do not turn on". The option governs whether relevant object visibility of layout view are turned on when errors are displayed. This is similar to checking or unchecking the *Auto Set Object Visible* menu item in the 'Options' menu button of the Error Browser. By default, layout object visibility will be turned on for related error object display.

`-auto_set_object_visible_type` *`SetVisibleType`*

The *-auto_set_object_visible_type* option accepts argument values *incremental*, or *exclusive* to mean turn on layers/objects incrementally or exclusively when a different error is selected. This is similar to selecting the *Auto Set Object Visible Incremental/Exclusive* menu item in the 'Options' menu button of the Error Browser. By default, layout object visibility will be turned on incrementally for related error object display.

Description

This command sets option setting for the Error Browser. Multiple option values can be set using one command.

Feedback

Examples

The following example sets the grouping of errors by layers, to show all errors, and to pan to selected errors.

```
prompt> gui_set_error_browser_option -grouping layer -show_mode all
 -view_mode pan
```

The following example sets the option to dim layout views when errors are displayed and to hide fixed errors:

```
prompt> gui_set_error_browser_option -dim true -hide_fixed true
```

See Also

- gui_error_browser
- gui_get_error_browser_option

---

gui_set_error_data_filter

Set a filter on open error data.

Syntax

string *gui_set_error_data_filter*

-show | -hide [-types *TypeList*] [-layers *LayerList*] [-objects *ObjectCollection*] [-typed_object_names *ObjectSpecification*] [-object_types *ObjectTypeSpecification*]

```
TypeList         String List
LayerList        String List
ObjectSpec       String List or collection
ObjectCollection collection
ObjectTypeList   String List
```

Arguments

`-show | -hide`

Mutually exclusive required option to specify whether the filtering should hide the records matching the filter criteria (-hide) or prune to the records matching the filter criteria (-show).

`-types` *`TypeList`*

Optional argument to specify a list of type names as filter criteria. An error record would match this criteria if its type attribute value matches any type name in the list.

`-layers` *`LayerList`*

Optional argument to specify a list of layer strings as filter criteria. An error record would match this criteria if its layer string matches any layer string in the list. The layer strings is the layer_names string attribute value for the error. It is is also displayed in the error browser.

`-objects` *`ObjectCollection`*

Optional argument to specify a set of design objects as filter criteria. An error record can be associated with zero or more design objects,such as nets, pins, and cells. An error record would match this criteria if any of its associated objects matches any object in the collection. Associated objects for an error can be accessed using get_attribute. The attribute name is tool specific. For tools which restrict the types of object that may be associated with errors, the attribute name may be specific, such as "nets". For tools that support a more general object association, the attribute name may be more general, such as "objects".

`-typed_object_names` *`ObjectSpecification`*

Optional argument to specify a set of design objects as filter criteria. An error record can be associated with zero or more design objects, such as nets, pins, and cells. An error record would match this criteria if any of its associated objects matches any objects in the specification. Specify objects with type-name, object-name-list pairs, for example

{ net { NetName1 NetName2 NetName3 {} } }.

Multilple such pairs may be listed to specify objects of multiple types as filter criteria, for example

{ { net { Net1 Net2 Net3 {} } } { pin { PinA PinB } } }.

Associated objects for an error can be accessed using get_attribute. The attribute name is tool specific. For tools which restrict the types of object that may be associated with errors, the attribute name may be specific, such as "nets". For tools that support a more general object association, the attribute name may be more general, such as "objects". A null object association can be specified using an empty string as the object name.

`-object_types` *`ObjectTypeList`*

Optional argument to specify a list of object type names as filter criteria. An error record would match this criteria if the type attribute value of any of its associated objects matches any type name in the list.

Specify object types with type-name, type-name-list pairs, for example

{ net_type { Signal Power } }.

Multilple such pairs may be listed to specify lists for multiple object types as filter criteria, for example

{ { net_type { Signal Power } } { route_type { Detail User } } }.

Description

Set the given filter on open error data. Filter state is set and is applied on currently open error data and subsequently opened error data until cleared. When a new filter is set, the new filter overrides any previously set filter. There is no accumulation of filters.

In order to show errors that have been hidden by a previously set filter, clear the filter with gui_clear_error_data_filter.

You can either filter out errors matching the filter specification (use -hide) or prune the current errors to errors matching the filter specification (use -show).

If multiple kinds of criteria are used in a filter, then an error must satisfy each kind of criteria to be a match. For example, if a filter with both an error type list (the -types option) AND an object type list (the -object_types option) is set, then for an error to be a match for the filter, its type must be one of the types set in the filter AND its associated object type must be one of the object types set in the filter.

Examples

The following example illustrates how to show only errors of type "Short":

gui_set_error_data_filter -show -types "Short"

The following example illustrates how to show only errors of type "Short" that are also associated with the "Signal" net type:

gui_set_error_data_filter -show -types "Short" -object_types {{net_type "Signal"}}

The following example illustrates how to hide errors associated with the "Global" route type:

gui_set_error_data_filter -hide -object_types {{route_type {Global}}}

The following example shows how to clear the error data filter:

gui_clear_error_data_filter

See Also

- gui_error_browser
- gui_clear_error_data_filter
- gui_set_error_browser_option

Feedback

gui_set_error_status

Set the error status value of one or more error records.

Syntax

string *gui_set_error_status*

-errors *Errors* -status *Status*

```
Errors            String
Status            Status
```

Arguments

`-errors` *Errors*

Errors specifies the handle for a collection of errors on which to set the fixed state.

`-status` *Status*

The *-status* argument specifies the status value. The supported status values are tool dependent. All tools with physical error data for the error browser support values "error" and "fixed". Some tools also support "ignored".

Description

This command sets the status of the given error objects to the given status value.

The status of an error is displayed in the Error Browser and an error can be hidden or shown based on its status value. The status is a part of the textual report written to a file by gui_report_errors.

See Also

- gui_error_browser
- gui_set_error_browser_option
- gui_report_errors

gui_set_hierview_data

Set various state in the current or named Hierarchy View and its sub-views.

Syntax

string *gui_set_hierview_data*

Feedback

```
[-view                  View_Name]
[-tree_type             Tree_Type]
[-tree_attr_group       Attr_Name]
[-tree_filter           String]
[-map_area              Attr_Name]
[-map_color             Attr_Name]
[-childlist_name        String]
[-childlist_attr_group Attr_Name]
[-childlist_filter      String]
[-elide                 Elide_Type]
View_Name      String
Attr_Name      String
Tree_Type      String (logical|rtl)
```

Arguments

`-view View_Name`

Use the given HierView window name (ex: Hier.1) for the following actions. Default is to get the most recent used HierView window name.

`-tree_type Tree_Type`

Set the current Hier Tree type to Logical or RTL hierarchy. RTL hierarchy is only available in RTLA. The default is logical.

`-tree_attr_group Attr_Name`

Set the current Hier Tree attribute group name.

`-tree_filter String`

Set the current Hier Tree filter expression.

`-map_area Attr_Name`

Set the Hier Map Area attribute group name.

`-map_color Attr_Name`

Set the Hier Map Color attribute group name.

`-childlist_name String`

Set the current ChildList sub-view name.

`-childlist_attr_group Attr_Name`

Set the current ChildList attribute group name.

`-childlist_filter String`

Set the current ChildList filter expression.

`-elide Elide_Type`

Set the text elide for the view. Supported types are 'middle', 'left', 'right'.

Feedback

Description

This command allows you to set various current Hierarchy View related view settings.

See Also

- gui_get_hierview_data

---

gui_set_highlight_options

Change the options that control highlighting.

Syntax

string *gui_set_highlight_options*

```
[-current_color color_id | -next_color | -auto_cycle_color enable]

string          color_id
bool            enable
```

Arguments

`-current_color color_id`

Changes the current highlight color to the specified value. The current highlight color is used as a default when no color is specified for some operations.

`-next_color`

Changes the current highlight color to the next color in the set of available colors. This will wrap around so that it can be used to cycle through the color set. The provided colors are yellow, orange, red, green, blue, purple, light_orange, light_red, light_green, light_blue, and light_purple.

`-auto_cycle_color enable`

Specify if the current color should automatically be incremented after each gui_change_highlight -add operation. The color is not advanced if an explicit color is specified with the -color option of gui_change_highlight.

Description

The gui_set_highlight_options command can be used to modify highlighting behavior.

Examples

Set the current highlight color to blue.

shell> gui_set_highlight_options -current_color blue

Feedback

Enable auto cycling.

shell> gui_set_highlight_options -auto_cycle_color true

Advance to the next color.

shell> gui_set_highlight_options -next_color

See Also

- gui_change_highlight
- gui_get_highlight
- gui_get_highlight_options

---

gui_set_hotkey

Sets a key binding to a Tcl command or a menu command in a GUI window.

Syntax

string *gui_set_hotkey*

```
-hot_key key_name
-tcl_cmd command_name | -menu menu_name
[-replace]
[-delete]
[-replay_log_only]
[-window_type window_type_name | -all_window_types]

key_name              string
command_name          string
menu_name             string
window_type_name      string
```

Arguments

`-hot_key key_name`

Specifies the key that you are associating with the specified command. The *key_name* is a string that contains the key name can also contain one or more modifiers. The valid modifier keys are *SHIFT*, *ALT*, and *CTRL*. You specify the modifier names by prepending them to the key name and connecting them with a plus sign (+).

If you bind an unmodified key and the shift-modified key is not bound, the binding works for both the shifted and nonshifted keys. If you set separate bindings for the shift-modified key and the unmodified key, the bindings are unique. Note that shift modifiers work only with letter keys and function keys.

Feedback

Menus display the first key binding defined for a menu item. The key binding always appears in uppercase with its modifiers. For example, an unmodified accelerator for the letter "a" is shown as "A' and a shift-modified "a" (an uppercase A) is shown as "SHIFT+A."

`-tcl_cmd` *`command_name`*

Specifies the tool command language (Tcl) code that the tool executes when you press the key and its modifiers, if any.

The *-tcl_cmd* option and the *-menu* option are mutually exclusive.

`-menu` *`menu_name`*

Specifies the menu command that the tool invokes if you press the key and its modifiers, if any, when the menu command is enabled.

The *menu_name* is a string that specifies the hierarchy of the menu labels, with the labels connected by arrows formed from a hyphen and a greater than sign (->). For example, "File->Exit" specifies the Exit command on the File menu. The *menu_name* can also include the mnemonic for the menu text, which is specified with an embedded ampersand (&), but they are not required to do so.

The *-menu* option and the *-tcl_cmd* option are mutually exclusive.

`-replace`

Replaces an existing key binding for the specified menu command or Tcl command with the specified key.

`-delete`

Removes the key binding from the specified menu command or Tcl command.

`-replay_log_only`

Limits logging of the Tcl command associated with the key to the command replay log file. The tool suppresses the command echo and result display to the xterm and console, and the command does not appear in the output log or the Tcl history log.

The tool ignores this option if you do not specify the *-tcl_cmd* option.

`-window_type` *`window_type_name`*

Specifies the type of top-level window that the key should apply to. (A top-level window is a window with a menu bar.) If you do not specify this option, the tool uses the application default window type.

The read-only variable *gui_default_window_type(3)* specifies the application default window type.

Feedback

`-all_window_types`

Sets the key binding in every type of window for which it is applicable.

For menu command key bindings, the tool searches all of the top-level windows for the specified menu command and adds the key binding in any window where it exists. If the tool does not find any applicable windows, it issues a warning.

For Tcl command key bindings, the tool adds the binding in every type of top-level window.

Description

This command binds a given key to a function in the graphical user interface (GUI). The function might be available on a menu or specified by Tcl code. A menu command can have multiple key bindings. Key bindings can be specified for built-in or user-defined menu commands.

Key bindings for menu commands have several additional features that Tcl command key binding do not have. The function for a menu command key binding is invoked only when the menu command is enabled. This ensures that the function is invoked only when it is valid to invoke the function. In addition, the primary key bindings for menu commands appear with the menu text to make it easier for you to remember the bindings.

Examples

The following example creates a menu item and adds an additional key binding to it.

```
prompt> gui_create_menu -menu "&Test->Sub&Menu->Echo Hi" \
   -tcl_cmd "echo hi" -hot_key "Ctrl+5"
prompt> gui_set_hotkey -menu "Test->SubMenu->Echo Hi" \
   -hot_key "Ctrl+Y"
```

The following example creates a key binding, P, that invokes a Tcl command to print a query on the currently selected objects.

```
prompt> gui_set_hotkey -tcl_cmd {query_objects [get_selection]} \
   -key P
```

See Also

- gui_report_hotkeys
- gui_create_menu
- gui_delete_menu
- gui_create_toolbar
- gui_delete_toolbar
- gui_create_toolbar_item

Feedback

- gui_delete_toolbar_item
- gui_default_window_type
- gui_custom_setup_files

---

gui_set_layout_layer_visibility

Set the visibility of layers in a layout window

Syntax

```
status gui_set_layout_layer_visibility
collection_of_layers
[-window windowID]
-toggle | -only
```

Data Types

```
string windowID
```

Arguments

collection_of_layers

The collection of layers or list of layer names provided in technology file.

-window *windowID*

This specifies the window to have its mode changed. If not specified then the currently active window will be used.

-toggle

Toggle the visibility of the specified layers. The option is mutually exclusive with -only.

-only

Turn off all layers and then turn on only the specified layers. The option is mutually exclusive with -toggle.

Description

This command is used to set visibility of layers in a layout view.

Examples

To turn on visibility of METAL2, METAL3 layers only:

```
shell-t> gui_set_layout_layer_visibility {METAL2 METAL3} -only -window
 Layout.1
1
```

Feedback

To toggle visibility of layer METAL3:

```
shell-t> gui_set_layout_layer_visibility {METAL3} -toggle -window
 Layout.1
1
```

See Also

- gui_set_setting

---

gui_set_layout_user_command

Set user defined command for layout input.

Syntax

status *gui_set_layout_user_command*

```
-apply_cmd TclCmd | -clear
[-input_type InputType]
[-snap_type SnapType]
[-status_text string]
[-cancel_cmd TclCmd]
```

Data Types

```
TclCmd    string
InputType one of [rectangle | line | polygon | point]
SnapType  one of [litho | site | midsite | wiretrack | midwiretrack |
 user]
```

Arguments

`-apply_cmd` *TclCmd*

The user procedure to be called when input completed. The coordinates will be passed as an argument to this procedure. The options is mutually exclusive with -clear.

`-clear`

Cancel all pending input and return layout to default mouse mode. The options is mutually exclusive with -apply_cmd.

`-input_type` *InputType*

The desired input type: rectangle | line | polygon | point. Default is rectangle

`-snap_type` *SnapType*

The desired snap type: litho | site | midsite | wiretrack | midwiretrack | user Default is litho

Feedback

`-status_text` *string*

The help string to be shown in layout window status bar.

`-cancel_cmd` *TclCmd*

The user procedure to be called when input is canceled.

Description

This command is used to facilitate creation of user defined tools to extend the layout view standard capabilities. The command sets layout view to given input mode and executes user callback procedure on input complete. The input points are passed as an argument to user callback.

Examples

Cut the object shape with user specified rectangle

```
proc my_cut_object_shape_proc { rect } {
  change_selection [cut_objects [get_selection] -bbox {$rect}]
}

gui_set_layout_user_command -apply_cmd {my_cut_object_shape_proc}
 -status_text "Drag the rectangle to cut selected objects" -input_type
 rectangle
1
```

See Also

- change_selection
- get_selection

---

gui_set_map_option

Sets the value for an option in the specified visual mode or map mode.

Syntax

```
string gui_set_map_option
-map map_name
-option option_name
-value value | -default
```

Data Types

```
map_name        string
option_name     string
value           string
```

Feedback

Arguments

`-map` *`map_name`*

Specifies the name of the visual mode or map mode.

`-option` *`option_name`*

Specifies the name of the option to be set.

`-value` *`value`*

Specifies the value for the option.

The *-value* option is mutually exclusive with the *-default* option. You must specify one, but not both.

`-default`

Sets the option to its default value.

The *-default* option is mutually exclusive with the *-value* option. You must specify one, but not both.

Description

This command sets the value for an option in a visual mode or map mode. You must specify the visual mode or map mode name and the option name. You must set the option to a specific value or its default value. Use the *-value* option to set a specific value or the *-default* option to set the option to its default value.

Examples

The following example sets the number of bins to 9 in the global route congestion map:

```
prompt> gui_set_map_option -map AREAPARTITION \\
        -option num_bins -value 9
```

See Also

- gui_get_bucket_option
- gui_get_bucket_option_list
- gui_get_map_list
- gui_get_map_option
- gui_get_map_option_list
- gui_set_bucket_option

Feedback

gui_set_mouse_tool_option

Set an option on a mouse tool

Syntax

string *gui_set_mouse_tool_option* -tool *string*

-option *string* -value *string*

```
string string
string string
string string
```

Arguments

`-tool string`

Tool to set option for.

`-option string`

Option to set.

`-value string`

New option value.

Description

This command sets a mouse tool option. Mouse tool options influence the operation of the mouse tool when that tool is active

Examples

```
> gui_set_mouse_tool_option -tool SELECT -option InputMode -value
 Rectangle
```

See Also

- gui_mouse_tool
- gui_get_mouse_tool_option

gui_set_performance_log_option

Configures performance log behavior of logging specific commands or all default commands

Feedback

Syntax

```
status gui_set_performance_log_option
[-commands command_names | -log_all]
[-add | -remove]
```

Data Types

command_names	list

Arguments

`-commands` *command_names*

This option is mutually exclusive -log_all. If -add or -remove are not given, the argument sets or refreshes the commands to be logged. If a logging process has not been started, this argument sets the specific commands to be logged for the upcoming logging process. If already been started, this argument refreshes the commands to be logged from now on.

If -add or -remove are given, the argument set the commands to be added to or removed from already existing commands list.

`-add`

This option is mutually exclusive with -remove and must be used with -commands. Use -add to add commands to the already existing commands list which is set by this command. If there is no existing commands list, this argument generates a commands list to be logged.

`-remove`

This option is mutually exclusive with -add and must be used with -commands. Use -remove to remove commands from the already existing commands list which is set by this command.

`-log_all`

This option is mutually exclusive with -commands. This option resets the commands to be logged as default.

Description

This command controls the performance log behavior of logging specific commands or all default commands. After execute option -commands, the default commands list is not effective until execute option -log_all. The commands to be logged can be reset as default by option -log_all. This command can be used before and during a logging process and change the commands to be logged. After successfully execute this command, the logging behavior changes.

Feedback

Examples

```
Configure log behavior which logs performance data before and after
 executing
commands of gui_zoom and win_select_object.

prompt> gui_set_performance_log_option -commands {gui_zoom
 win_select_objects}

Add command gui_get_current_window to the already existing commands list.

prompt> gui_set_performance_log_option -commands {gui_get_current_window}
 -add

Remove command gui_get_current_window from the already existing commands
 list.

prompt> gui_set_performance_log_option -commands {gui_get_current_window}
 -remove

Reset the log behavior which logs performance data of all default
 commands.

prompt> gui_set_performance_log_option -log_all
```

See Also

- gui_log_performance
- gui_get_performance_log_option

---

gui_set_pref_value

Set the value of a preference key.

Syntax

string *gui_set_pref_value* -key *Key* -value *Value*

```
[-category Category]

Key             String
Value           Depends on the data type of the value of the specified
 key
Category        String
```

Arguments

```
-key Key
```

Key specifies the key to set the new value.

Feedback

`-value Value`

Value specifies the new value of the key. The new value should be of the same data type, one of (integer,bool,double,color,string) as the data type of the value of the key.

`-category Category`

Category specifies the category that owns the specified key. If not specified, a default category will be used.

Description

This command set the value of a preference key. Returns the value of the preference key.

Examples

The following example create a user-defined category *some_category* for storing preferences and then create a boolean key *some_key* under that category with the initial value of false. The command *gui_set_pref_value* is then used to change the value of the preference key to true.

```
gui_create_pref_category -category some_category
gui_create_pref_key -category my_category -key some_key -value_type bool
 -value false
gui_set_pref_value -category my_category -key some_key -value true
```

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_set_preset

Sets the specified preset for object specified by category

Feedback

Syntax

```
status gui_set_preset
-category category
[-system]
[-shared]
-default
-name name
```

Data Types

```
category    string
name         string
```

Arguments

`-category category`

Specifies the name of the category that the presets be applied to.

`-system`

Apply given system preset. This option is mutually exclusive with the -shared, -default option.

`-shared`

Apply given shared preset. This option is mutually exclusive with the -system, -default option.

`-default`

Apply preference setting. This option is mutually exclusive with the -system, -shared, preset option.

`-preset preset`

Specifies the preset to be applied. This option is mutually exclusive with the -default option.

Description

This command applies the default or specified preset to a category.

Examples

The following example applies the system presets 'floorplan' to the Layout.

```
prompt> gui_set_preset -category Layout -name floorplan -system
```

See Also

- gui_get_presets

Feedback

gui_set_region

Sets the coordinates of the current region rectangle or rectilinear polygon.

Syntax

```
status gui_set_region
shape
```

Data Types

```
shape            list
```

Arguments

shape

Specifies the region coordinates.

For a rectangle, the values should be given in the following format:

```
{{x1 y1} {x2 y2}}
```

For a rectilinear polygon, the values should be given in the following format:

```
{{x1 y1} {x2 y2} ... {xN yN}}
```

Description

This command sets the region shape to be used in layout region mode and returns 1 if it is successful.

Examples

```
prompt> gui_set_region {{-1743.945 18.357} {42.833 1523.657}}
1
```

See Also

- gui_get_region

gui_set_selected_errors

Changes the selection in the Error Browser to add or replace errors.

Syntax

int *gui_set_selected_errors*

```
-add | -replace errors
```

Feedback

Data Types

errors string

Arguments

-add *errors*

Adds the specified errors to the current selection. This option is mutually exclusive with the *-replace* option.

-replace *errors*

Replaces the specified errors in the current selection. This options is mutually exclusive with the *-add* option.

Description

Sets new selected errors or adds new selected errors in the Error Browser. Errors are identified as a collection of error objects. If the *-add* option is given, the command adds the given errors to the currently selected errors. If the *-replace* option is given, the command replaces the current selection with the given errors. The command fails if the given errors are not in the current set specified by the *gui_set_current_errors* command, or with a click in the Error Browser tree view. The command returns 1 if successful, 0 otherwise.

See Also

- gui_clear_selected_errors
- gui_error_browser
- gui_set_current_errors

---

gui_set_setting

Sets a setting on the specified window.

Syntax

gui_set_setting

```
-window WindowID
-setting Setting
-value Value

WindowID        String
Setting        String
Value        String | Bool | Int | Double
```

Feedback

Arguments

`-window WindowID`

WindowID specifies the window to set the setting

`-setting Setting`

Setting specifies the setting to set

`-value Value`

Value specifies the value of the setting

Description

Sets a setting on the specified window. Windows are views or top level windows. The setting is a property of the window and the value's type is specific to the given setting.

Examples

```
shell> gui_set_setting -window Layout.1 -setting showCell -value true
```

```
shell> gui_set_setting -window Layout.1 -setting viewLevel -value 1
```

See Also

- gui_get_setting

---

gui_set_task_list

Set the list of tasks that are visible in the task assistant, and set their order.

Syntax

gui_set_task_list

```
-tasks    TaskList

TaskList    List
```

Arguments

`-tasks TaskList`

TaskList specifies the list of tasks that will be visible in the task assistant and their order. Tasks which are not included in the list will still exist but will not be shown in the selection combo box of the task assistant.

Description

The gui_set_task_list command sets the list of tasks that are visible and available in the task assistant and sets their order.

Feedback

Examples

The following simple example sets the visible list of tasks to "Design Planning" followed by "Routing".

```
gui_set_task_list -tasks {{Design Planning} {Routing}}
```

See Also

- gui_get_task_list
- gui_create_task

---

gui_set_user_units

Sets the units for GUI.

Syntax

```
status gui_set_user_units
-type unit_type
-value unit_value
```

Data Types

```
unit_type          string
unit_value         string
```

Arguments

`-type unit_type`

Specifies the type of unit to set. The valid value is: "power".

`-value unit_value`

Specifies the unit value. The format of *unit_value* is "positive double number + *unit*". For power, examples include "1mW", "1e-2W", "7.1uW", and so on.

Description

This command sets the units that are used for GUI. "power" is the sole option for *unit_type* currently.

Examples

The following example sets the unit of power to 120mW.

```
prompt> gui_set_user_units -type power -value 120mW
1
```

Feedback

See Also

- gui_get_user_units

---

gui_set_utable

Set various state about User Tables.

Syntax

string *gui_set_utable*

```
[-name          Table_Name]
[-parent        Table_Name]

[-tag           Report_Tag]
[-title         Report_Title]
[-comment       Report_Comment]
[-menu          Root_Menu]

[-prefix        Prefix_Char [-column Column_Name] ]

[-action        TCL_Expr  -column Column_Name  -link Link_Text]

[-create_hook   TCL_Proc]
[-close_hook    TCL_Proc]

[-enter_hook    TCL_Proc]
[-exit_hook     TCL_Proc]

[-select_hook   TCL_Proc]
[-sel_action    TCL_Proc]
[-dsel_action   TCL_Proc]

[-window        Window_name]
[-filter        Filter_Expr]

Table_Name          String
Report_Title        String
Report_Comment      String
Report_Tag          String
Root_Menu           String
TCL_Proc            TCL procedure call
Column_Name         String
Link_Text           String
Prefix_Char         Chars(afpnumkMGTPE), "A" for Auto, " " for none.
Window_Name         String
Filter_Expr         String
```

Feedback

Arguments

`-name` *`Table_Name`*

A table name argument used with other arguments below.

`-parent` *`Table_Name`*

Set the parent table name for the given table name argument.

`-tag` *`Report_Tag`*

Set a tag string for the given table name argument.

`-title` *`Report_Title`*

Set a report title for the given table name argument.

`-comment` *`Report_Comment`*

Set a report comment for the given table name argument.

`-menu` *`Root_Menu`*

Assign a TCL defined Context Sensitive Menu Root name to this table that is shown when the user right clicks on any table cell in the table.

`-column` *`Column_Name`*

The column name argument is used with other arguments below.

`-prefix` *`Prefix_Char`*

Formats numeric data to the given prefix char. Special prefix characters are " " space which means no formating and "A" which means automatic formating. If given a table name, then it only applies to that table. If also given a column name, then it only applies to that tables column. If given by itself then it sets the default for all table columns that are numeric. Examples: (...,n=nano,m=milli,k=killo,M=Mega...)

`-link` *`Link_String`*

The link string use with the *-action* option below.

`-action` *`TCL_Expr`*

These options allow you to define a link action for every empty table cell in a given column. A link action is basically a TCL procedure call. In addition you can pass arguments that reference other values in the same row but in different columns. This option required the use of the options *-column* which indicates the column that this action is applied to and the *-link* text that is shown in the table cell under that column. (See Example Below)

Feedback

`-create_hook` *`TCL_Proc`*

Set table create TCL hook proc name. The proc is called when a new UserTable is created.

`-close_hook` *`TCL_Proc`*

Set table close TCL hook proc name. The proc is called when a UserTable is closed.

`-enter_hook` *`TCL_Proc`*

Set a specific table name enter hook TCL proc name. The named proc is called when a UserTable with the specified *-name* option is entered/shown. The argument to the proc is the name of the table that was entered/shown. Note: The enter hook is only run if the table is changed for the current UserTable window.

`-exit_hook` *`TCL_Proc`*

Set a specific table name exit hook TCL proc name. The named proc is called when a UserTable with the specified *-name* option is exited to show another table. The argument to the proc is the name of the table that was exited. Note: The exit hook is only run if the table is changed for the current UserTable window.

`-select_hook` *`TCL_Proc`*

Set table data select TCL hook proc name. This proc is called when any Table cell in any column is sected.

`-sel_action` *`TCL_Proc`*

Set the TCL action proc name that is called when a selection is made in the *-column* option given above. The arguments to the proc are the data of the selection and the selected row number and the total number of rows selected.

`-dsel_action` *`TCL_Proc`*

Set the TCL action proc name that is called when a deselection is made in the *-column* option given above. The arguments to the proc are the data of the deselection and the deselected row number and the column name of the new selection. An empty new column argument means a selection in the same column was made.

`-window` *`Window_Name`*

This option is used with other options below.

`-filter` *`Filter_Expr`*

This allows you to set the filter value on the current table visible in the view given by the '-window' option.

Feedback

Description

This command allows you to set various state and actions on User Tables.

Examples

The following example creates a new table MyTable and assigns the various state
and action options described above.

```
        # Create MyTable with columns: cell, voltage, current and add
 some data
# Note: the last column "Show" is left empty for an action column.
gui_create_utable -name MyTable \\
  -columns {cell voltage:double current:double Show}
gui_append_utable -name MyTable -rows {
  {A 5.0 0.2 {}}
  {B 6.0 0.4 {}}
  {C 7.0 0.6 {}}
}

# Create a context menu with one item:
set menuRoot "My Custom Menu"
gui_create_menu -root $menuRoot -menu "MyItem" -tcl_cmd "echo MyItem"

# Create various hooks that are executed at create/close and selection.
proc createTableHook { table } {
  puts stdout "*New table created: $table"
}
proc closeTableHook { table } {
  puts stdout "*Closing table: $table"
}
proc selectHook { table } {
  set win [gui_get_current_window -mru -type UserTable]
  set columns [gui_get_utable -window $win -columns]
  set values  [gui_get_utable -window $win -values $columns]
  foreach v $values {
    echo "Table: $table, Selected Value: $v"
  }
  # return "1" if default selection action should be ignored
  return "0"
}
proc selAction {data row rowCount} {
  puts stdout "selAction $data $row $rowCount"
}
proc dselAction {data row newColumn} {
  puts stdout "dselAction $data $row $newColumn"
}

# Create an action proc used for a column action
```

Feedback

```
proc power { v c } {
  puts stdout [expr $v * $c]
}

# Set various state and actions on my table:
gui_set_utable -name MyTable \\
    -title "My Report" \\
    -tag   "My Tag" \\
    -comment "This is a comment..." \\
    -menu $menuRoot \\
    -create_hook createTableHook \\
    -close_hook  closeTableHook \\
    -sel_action selAction \\
    -dsel_action dselAction \\
    -select_hook selectHook \\
    -action {power @{voltage} @{current}} \\
      -column Show -link Power


# Note the action calls the TCL proc power and also references
# column values using the @{column_name} argument syntax.

gui_show_utable -name MyTable -mru
```

See Also

- gui_create_utable
- gui_export_utable
- gui_foreach_utable_row
- gui_get_utable
- gui_import_utable
- gui_set_utable_meta
- gui_set_utable_values
- gui_show_utable

---

gui_set_utable_meta

Create a MetaData Object for use in creating a new UserTable.

Syntax

string *gui_set_utable_meta*

Feedback

```
-name         MetaDataObj_Name
[-remove]
[-title       Report_Title]
[-comment     Report_Comment]
[-tag         Report_Tag]
[-menu        Root_Menu]
[-col_name    Column_Name]
[-col_type    Column_Type]
[-read_only]
[-hidden]
[-user_enums Column_Type_Enums]
[-user_list  Column_Type_List]

MetaDataObj_Name   String
Report_Title       String
Report_Comment     String
Report_Tag         String
Root_Menu          String
Column_Name        String
Column_Type        String
Column_Type_Enums  List of Strings
Column_Type_List   List of Strings
```

Arguments

`-name MetaDataObj_Name`

The name of the MetaData object.

`-remove`

Remove and free the given MetaData object.

`-title Report_Title`

Report title for a table.

`-comment Report_Comment`

Report comment for a table.

`-tag Report_Tag`

A tag string assigned to the table.

`-menu Root_Menu`

User Context Sensitive Menu Root name.

`-col_name Column_Name`

A column name that has the following column type.

`-col_type Column_Type`

A column type name.

-read_only

Mark the given column name as read only which prevents editing data in that column.

-hidden

Mark the given column name as hidden.

-user_enums *Column_Type_Enums*

A list of single string values for a new user type defined by the -col_type argument.

-user_list *Column_Type_List*

A list of multi string values for a new user type defined by the -col_type argument. A table cell of this type can contain multiple unique values of this type separated by a comma.

Description

```
This command creates a new MetaData object that then can be used to add
 useful meta data to
the creation of a new UserTable. A new UserTable is created via the
 gui_create_utable
and the gui_import_utable commands. MetaData can be used to define
 certain column data/object
types instead of using column name postfixes. Users can also define new
 data types that
are made of a list of unique string values. A table cell can then be set
 to only allow
single values (enums) of that new type or a comma separated list of
 values (lists) of that
new data type. MetaData can also be exported via the gui_export_utable
 command along with
the table data.
```

Examples

```
The following example creates a new MetaData object named myMetaObj which
 contains a
report title and comment.

shell> gui_set_utable_meta -name myMetaObj -title "My Report" -comment
 "This is a comment"

The following example adds a definition of a new data type 'myData'
 assigned to
the column 'myDataColumn'. This new data type has 4 values (valA valB
 valC and valD).
Also optionally the user can add a help string after the value separated
 by a
colon ':' which is used to show tool tips.
```

Feedback

```
shell> gui_set_utable_meta -name myMetaObj -col_name myDataColumn
 -col_type myData -user_enums {
  {valA: Help string for valA...}
  {valB: Help string for valB...}
  {valC: Help string for valC...}
  {valD: Help string for valD...}
}

The following example then imports a CSV file and adds the previously
 defined MetaData to
that new table.

shell> gui_import_utable -file myTable.csv -meta_obj myMetaObj
```

See Also

- gui_create_utable
- gui_export_utable
- gui_get_utable
- gui_import_utable
- gui_set_utable
- gui_show_utable

---

gui_set_utable_values

Set the given UserTable values in the given column(s) and the filtered set of rows.

Syntax

string *gui_set_utable_values*

```
-name             Table_Name
-columns          Column_List
-values           Value_List
[-filter          Filter_Expr]

Table_Name      String
Column_List     StringList
Value_List      StringList
Filter_Expr     String
```

Arguments

`-name Table_Name`

The table name on which to set the new values.

Feedback

`-columns` *`Column_List`*

Provides a list of one or more column names that indicate the location of values to set.

`-values` *`Value_List`*

Provides a list of values used to set in the columns given above. *Note:* The number of values must match the number of columns.

`-filter` *`Filter_Expr`*

To only update a certain set of rows in the table with new values you must ether first filter the table in the view or use this filter expression to only target the rows of interest.

Description

```
This command allows you to update/change the values in the table rows
selected by the given columns and the given filter value.
If no filter option is given the all rows will be updated unless the
table is currently viewed, then all rows will be updated that match
the current view filter setting.
```

Examples

```
The following example updates the value of column 'direction' to 'out'
and the column 'state' to the value 'open' for all the rows that match
the filter where 'pin_module' is equal to 'my_module'.
```

```
shell> gui_set_utable_values -name $table -column {{direction} {state}}
 -value {out open} \\
     -filter {pin_module == my_module}
```

See Also

- gui_get_utable
- gui_set_utable
- gui_foreach_utable_row

---

gui_set_vm

Sets visual mode attributes.

Syntax

string *gui_set_vm*

`-name` *`identifier`*
`[-update_cmd` *`update_command`*`]`

```
[-title label]
[-help_topic subject]
[-infotip infotip]
[-buckets buckets]
[-icon string]
[-netfilter string]
[-float bool]
[-set_exaggerations]
[-top_exaggeration float]
[-mid_exaggeration float]
[-bot_exaggeration float]
[-show_only_pins_of_nets bool]
[-enable_bucket_delete bool]
```

Data Types

```
identifier          string
update_command      string
label               string
subject            string
infotip             string
buckets             string
net_filter          string
icon_spec           string
```

Arguments

`-name identifier`

Specifies the name of the visual mode to be modified. This name is used as the argument when the associated visual mode commands requires a visual mode name to be specified.

`-update_cmd update_command`

Specifies an optional Tcl command that is executed when the visual mode is accessed. A typical use for this command is to update the state of the visual mode if its contents are likely to change under certain circumstances.

`-title label`

Specifies an optional title. The title can include embedded spaces. Use this option to provide a label for the visual mode in the GUI.

If you do not specify the *-title* option, the default title is an empty string and the tool uses the *-name* option value to provide the label.

`-help_topic subject`

Specifies the help topic text string. This text is used as the subdirectory to find a help page related to the visual mode.

Feedback

`-infotip infotip`

Specifies the infoTip string.

`-buckets buckets`

Specifies the rendering order of the buckets. The tool renders the specified buckets from left to right based on their order in the list. Buckets that are not included in the list are rendered before any of the buckets that are specified in the list.

`-icon string`

Specifies the icon file name. Use this option to provide an icon for GUI menus.

`-netfilter string`

Specifies the net connection filtering.

`-float true | false`

Specifies whether the bucket contents have a floating point range. By default, the value is false.

`-set_exaggerations`

Specifies the bucket exaggerations.

`-top_exaggeration float`

Specifies the exaggeration for the top bucket, the value should be greater than or equal to zero. By default, the value is 0.

`-mid_exaggeration float`

Specifies the exaggeration for the middle bucket, the value should be greater than or equal to -1. By default, the value is -1.

`-bot_exaggeration float`

Specifies the exaggeration for the bottom bucket, the value should be greater than or equal to zero. By default, the value is 0.

`-show_only_pins_of_nets true | false`

Indicates that layout only show pins of net objects in buckets.

`-enable_bucket_delete true | false`

Specifies whether the buckets can be deleted from context menu. Only discrete visual mode buckets can be deleted. By default, the value is false.

Description

The *gui_set_vm* command modifies one or more attribute values of an existing visual mode.

Feedback

Examples

The following example changes the title of a visual mode from "mycoloring1" to "test":

```
prompt> gui_set_vm -name mycoloring1 -title {test}
```

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm
- gui_get_vmbucket
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vmbucket
- gui_update_vm

---

gui_set_vmbucket

Set attributes for Visual Mode Bucket

Syntax

string *gui_set_vmbucket* -vmname *mode_identifier*

-name *bucket_identifier* [-infotip *infotip*] [-netfilter *net_filter*] [-color *color*] [-pattern *pattern*] [-exaggeration double] [-number int] [-maxval double] [-minval double] [-visible *visibility*] [-title *label*] [-collection *handle*] [-above *bucket_identifier* | -below *bucket_identifier* | -at *top*| *bottom*]

```
string  mode_identifier
string  bucket_identifier
string  infotip
string net_filter
string  color
string  pattern
string  visibility
string  label
string  handle
string  bucket_identifier
string  bucket_identifier
string  top|bottom
```

Feedback

Arguments

`-vmname` *mode_identifier*

Specifies the visual mode to which the bucket is a member. The visual mode must already exist. To see the current list of defined visual modes use the gui_list_vm command.

`-name` *bucket_identifier*

Specifies the name of the visual mode bucket to be modified. To see the list of buckets associated with a specific visual mode use the gui_get_vm command with the -buckets option.

`-infotip` *infotip*

String for infotip.

`-netfilter` *net_filter*

Specifies net connection filtering.

`-color` *color*

Specifies bucket rendering color. Supports color naming by RGB triplet (for example: "#FFFF00" for yellow) as well as standard names (for example: "yellow").

`-pattern` *pattern*

Specifies bucket rendering fill pattern. Supported patterns are: SolidPattern, HorPattern, VerPattern, CrossPattern, BDiagPattern, FDiagPattern, DiagCrossPattern, Dense1Pattern, Dense2Pattern, Dense3Pattern, Dense4Pattern, Dense5Pattern, Dense6Pattern, Dense7Pattern, NoBrush.

`-exaggeration double`

Specifies bucket min pixel exaggeration value >= 0.

`-number int`

Specifies bucket display number value >= 0 that is used to provide a display number for the visual mode bucket in the gui. If the -number option is not specified, it will default to the number of objects in the bucket.

`-maxval double`

Specifies bucket maximum value.

`-minval double`

Specifies bucket minimum value.

`-visible` *visibility*

Specifies visibility state of bucket contents. Valid values are TRUE and FALSE.

`-title label`

Specifies an optional string (that can include embedded spaces) that is used to provide a label for the visual mode bucket in the gui. If the -title option is not specified, it will default to an empty string and the -name argument value value will be used to provide the labeling instead.

`-collection handle`

Tcl object collection to be colored by this visual mode bucket.

`-above bucket_identifier`

Specifies an optional visual mode bucket name above which the current bucket is to be rendered.

`-below bucket_identifier`

Specifies an optional visual mode bucket name below which the current bucket is to be rendered.

`-at top|bottom`

Specifies an optional string indicating whether the current visual mode bucket is to be rendered at the top or bottom of all other buckets. Valid values are top and bottom.

Description

The gui_set_vmbucket command modifies an instance of a visual mode bucket.

Examples

Change the color and pattern for the bucket named "cat1" in the visual mode "foo":

shell> *gui_set_vmbucket -vmname foo -name cat1 -color #00FF00 -pattern FDiagPattern*

Color the objects in the current selection red by placing them in the bucket named "cat1" in the visual mode "foo" and setting its bucket color:

shell> *gui_set_vmbucket -vmname foo -name cat1 -collection [get_selection] -color red*

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm
- gui_get_vmbucket
- gui_list_vm

Feedback

- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vm
- gui_update_vm

---

gui_set_window_pref_key

Create a preference key owned by a particular window or window type

Syntax

```
new_value gui_set_window_pref_key
{ -window window_id | -window_type window_type }
[ -category category ]
-key key
-value_type value_type
-value value
```

Data Types

```
window_id       string
window_type     string
category        string
key             string
value_type      string
value           string
```

Arguments

`-window window_id`

Specifies the name of the window that the preference will be applied to. This option is mutually exclusive with the -window_type option.

`-window_type window_type`

Specifies the name of the window type that instances of the type will have the preference be applied to. This option is mutually exclusive with the -window option.

`-category category`

Specifies the category that the key belongs to. If not specified, a default category will be assigned.

`-key key`

Specifies the name of the key to create.

Feedback

`-value_type` *value_type*

Specifies the the data type of the value of the key. It must be one of [bool | integer | double | color | string].

`-value` *value*

Specifies the value of the key. The value should be set appropriately in accordance with its value type. The bool type accepts [true, false, 0, 1, on, off]. The color type accepts a named color, eg. "red" or a string specifying the color in this format "#RRGGBB", for example, "#0000FF".

Returns

new_value

The new value of the key.

Description

This command creates a preference key with the specified key name or sets the value of an existing pref key. This key will have the specified value type and value, and it will be created under the specified category. If the category is not specified, the key will belong to a default category. Returns the value of the preference key.

Examples

The following example will create a window and set a preference on that window.

```
shell> set wnd [gui_create_window -type Console]
shell> gui_set_window_pref_key -window $wnd -category {caption} -key
 {title}
 -value_type string -value {Command Console}
```

See Also

- gui_get_window_pref_value
- gui_get_window_pref_categories
- gui_get_window_pref_keys
- gui_create_pref_category
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories

Feedback

- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key
- gui_update_pref_file

---

gui_set_window_preset

Sets the specified preset for object specified by window_type

Syntax

```
status gui_set_window_preset
-window_type window_type
[-system]
[-shared]
-default
-preset preset
```

Data Types

```
window_type     string
preset          string
```

Arguments

`-window_type window_type`

Specifies the name of the window type that the presets be applied to.

`-system`

Apply given system preset. This option is mutually exclusive with the -shared, -default option.

`-shared`

Apply given shared preset. This option is mutually exclusive with the -system, -default option.

`-default`

Apply preference setting. This option is mutually exclusive with the -system, -shared, preset option.

`-preset preset`

Specifies the preset to be applied. This option is mutually exclusive with the -default option.

Feedback

Description

This command applies the default or specified preset to a window type. This command has been deprecated, please use gui_set_preset instead.

Examples

The following example applies the system presets 'floorplan' to the Layout.

```
prompt> gui_set_window_preset -window_type Layout -preset floorplan
 -system
```

See Also

- gui_set_preset
- gui_get_presets
- gui_get_window_presets

---

gui_show_clock_network

Brings up the dialog for the PrimeTime GUI command gui_show_clock_network

Syntax

status *gui_show_clock_network*

```
[-clock_network]
[-clock_source_network]
[-all]
[-used_as_clock]
[-used_as_data]
[-both]
```

Arguments

`-clock_network`

Takes into consideration only the clock_network for the selected clocks.

`-clock_source_network`

Consideres the clock_source_latency_network for the selected clocks.

`-all`

Takes into consideration total clock_network for the selected clocks. This is same as specifying -clock_network and -clock_source_network

`-used_as_clock`

Consideres the clock pins which are only used as clocks.

Feedback

`-used_as_data`

Consideres the clock pins which are only used as data.

`-both`

Tis takes into onsideration all the pins (pins used as clocks and data).

Description

This command brings up the dialog for the command gui_show_clock_network. Make sure you have a valid clock on the selection bus before issuing this command, else the command will not process the choosen options.

Examples

The following example brings up the dialog

```
pt_shell> change_selection [get_clocks]

pt_shell> gui_show_clock_network -all -both
```

See Also

- change_selection

---

gui_show_command_form

Generate and show a form dialog for a shell command.

Syntax

string *gui_show_command_form* -command *command*

```
string command
```

Arguments

`-command` *command*

The name of the shell command to show in the form.

Description

Create a form dialog for a shell command. The form contains fields to allow entry of option arguments.

The dialog provides the following operations for the command:

```
- execute the command
- append the command to a script
- show the man page for the command.
```

Feedback

This command creates form dialogs for application defined commands. It will also create a form dialog for a public user defined tcl proc when its option value types have been defined using the *define_proc_attributes* command. The kinds of value input fields used in the form depends on the *type_name* attribute given for the option using the *-define_args* option. See man pages for *define_proc_attributes* and *gui_report_proc_arg_type_names* for more information.

This command is available only when the GUI is running.

Examples

The following example creates a form dialog for the command *gui_start*

```
shell> gui_show_command_form -command gui_start
1
```

The following is an example of a tcl proc definition followed by a *define_proc_attribute* command invocation to define its attributes. The proc is then shown in a form dialog with the *gui_show_command_form* command. The example proc accepts a net and routing layers as arguments.

```
shell> proc myProc {args} {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo $argname = $results($argname)
  }
}
shell> define_proc_attributes myProc \\
  -info "echo argument values" \\
  -define_args { \\
    {-net "select a net" "net" string \\
          {required {type_name net}}} \\
    {-layers "select routing layers" "routing layers" list \\
          {optional {type_name {layer {subtype routing}}}}} \\
  }
shell> gui_show_command_form -command myProc
```

See Also

- define_proc_attributes
- gui_report_proc_arg_type_names
- gui_start
- gui_stop

---

gui_show_file_in_editor

Show the contents of a file in an external text editor.

Feedback

Syntax

```
gui_show_file_in_editor
-filename filename
[ -line linenum ]
```

Data Types

```
filename string
linenum  integer
```

Arguments

`-filename filename`

filename specifies the name of the file to be edited.

`-line linenum`

linenum specifies the line in the file to start editing at.

Description

The gui_show_file_in_editor command opens a text editor to edit the specified file. The editor is a separate task so doen not block the current application.

By default the file is edited using the text editor from the EDITOR environment variable in a xterm. If the EDITOR environment variable is not set then the 'vi' text editor is used.

Both the editor and terminal (xterm) are searched for on the users path. If either the editor or terminal can no be found the command will fail.

Examples

The following example shows the file "test.txt" starting at line 100.

```
shell> gui_show_file_in_editor -file test.txt -line 100
```

See Also

- gui_show_url_in_browser

---

gui_show_man_page

Show a man page in the man browser

Syntax

string *gui_show_man_page* [-apropos]

[*topic*]

```
string topic
```

Feedback

Arguments

topic

Man page topic to be shown.

-apropos

Use the apropos command so search for commands that are related to the given topic.

Description

This command will launch or raise the man page browser, and display the topic specified in it. If no topic is specified, then the HOME page for man page indexes will be shown.

If the specified topic does not exactly match a given man page, then additional searching will be done for commands. If the topic matches one or more commands, then an index page will be generated showing all of the commands which match the specified topic pattern. If the topic doesn't match any commands, then we will use the topic as the initial part of a command and add a * to it and do command matching. Any completions for the command will be shown on an index page that can then be used to select a man page for viewing.

If the -apropos option is specified, then the topic will be searched via the apropos command and the topics returned will be shown with thier brief help strings, and provide links the the man pages referenced.

Examples

To show the man page for the find command:

gui_show_man_page find

To show the man page for all commands containing the word cell:

gui_show_man_page *cell*

To show the man page for the get_attribute command using command completion:

gui_show_man_page get_attr

See Also

- man
- apropos

---

gui_show_map

Displays or hides the specified visual mode or map mode.

Feedback

Syntax

```
string gui_show_map
-map map_name
-show true | false
[-window window]
```

Data Types

```
map_name        string
window          string
```

Arguments

`-map map_name`

Specifies the name of the visual mode or map mode.

`-show true | false`

Displays the visual mode or map mode when set to *true*, or hides the visual mode or map mode when set to *false*.

`-window window`

Specifies the name of layout window in which to display or hide the visual mode or map mode.

Description

This command displays or hides the specified visual mode or map mode. If the *-window* option is not specified, the command uses the most recently active layout window.

Examples

The following example displays the highlight visual mode in the most recently active layout window:

```
prompt> gui_show_map -map Highlight \\
        -show true
```

See Also

- gui_get_map_list

---

gui_show_palette

Shows the palette in the specified window state.

Syntax

```
status gui_show_palette
{ -name palette_id | -type palette_type }
```

Feedback

```
[ -parent parent_name ]
[ -show_state window_state ]
[ -dock_edge dock_edge ]
[ -size {width height} ]
```

Data Types

```
palette_id    string
palette_type  string
parent_name   string
window_state  string
dock_edge     string
width         float
height        float
```

Arguments

`-name palette_id`

Specifies the palette name id or title.

If the *-parent* option is specified only palettes in the specified parent toplevel window are shown. If the *-parent* option is not specified all palettes of this name in all toplevel windows are shown.

This option precludes the *-type* option

`-type palette_type`

Specifies the palette type id. Any matching palette of this type will be shown.

If the *-parent* option is specified only palettes in the specified parent toplevel window are shown. If the *-parent* option is not specified all palettes of this type in all toplevel windows are shown.

This option precludes the *-name* option

`-parent parent_name`

Specifies the parent toplevel window name.

`-show_state window_state`

Specifies the window state to show the palette.

Legal states are *maximized*, *minimized*, *normal*, or *docked*.

The default is *normal*.

`-dock_edge dock_edge`

Specifies the edge to dock the palette.

Legal dock edges are *left*, *top*, *right*, or *bottom*.

Feedback

Note: this option is only valid if the state is set to *docked*.

The default is *right*.

`-size {width height}`

Specifies the width and height of the palette.

Note: this option is only valid if the state is set to *normal*.

Description

This command shows the specified palette in the specified state and optionally sets the geometry.

If no state is specified then it defaults to normal.

Examples

The following example shows the Layer palette docked on the left of the workspace.

```
shell> set top [gui_create_window -type TopLevel]
shell> set layout [gui_create_palette -type Layout -parent $top]
shell> gui_show_palette -name $layout -show_state docked -dock_edge left
```

See Also

- gui_hide_palette

---

gui_show_parasitics

Highlights parasitics for a specified set of nets.

Syntax

status *gui_show_parasitics*

```
[-parasitic_corners corner_names]
[-all_parasitic_corners]
[-aggressor_net aggr_net]
[-nores]
[-nocg]
[-nocc]
[-novia]
nets
```

Data Types

aggr_net	string
corner_names	string
nets	string

Feedback

Arguments

`-parasitic_corners` *corner_names*

A list of one or more corner names for which to display parasitic element values.

`-all_parasitic_corners`

Shows values from all corners.

`-aggressor_net` *aggr_net*

Specifies an aggressor net for which to show coupling capacitance.

`-nores`

Do not display parasitic resistors.

`-nocg`

Do not display parasitic ground capacitors.

`-nocc`

Do not display parasitic coupling capacitors.

`-novia`

Do not display via parasitics.

nets

A list of one or more nets for which to show parasitics. At least one net is required. Glob-style wildcards (*) are supported.

---

gui_show_rtl_source_file_line

Show the contents of an RTL file in the RTL Source Browser view.

Syntax

```
status gui_show_rtl_source_file_line
[ -window string ]
-filename string
[ -line int ]
```

Arguments

`-window` *string*

Window specifies the GUI window to show the view in.

This option is optional, the current window is used by default.

Feedback

`-filename` *string*

Filename specifies the name of the file to be show.

`-line` *int*

Line specifies the line in the file to move to.

This option is optional, the first line of the file is used by default.

Description

The gui_show_rtl_source_file_line command shows the contents of the specified RTL file in a RTL Source Browser view. A line can also be specified to automatically move to that line in the file.

If the format of the file is recognised then syntax highlighting will be added and folding of blocks allowed.

Currently Verilog, System Verilog and VHDL files are supported and are recognised by their file suffix (.v, .sv/.vs, .vhd/.vhdl).

Examples

The following example shows the file "test.v" starting at line 100.

```
shell> gui_show_rtl_source_file_line -filename test.v -line 100
```

See Also

- gui_create_window

---

gui_show_short_regions

Displays noncritical polygons, including metal fill polygons, in the vicinity of a short identified by the StarRC tool during extraction.

Syntax

status *gui_show_short_regions*

`[-gpd` *gpd_dir*`]`

Data Types

gpd_dir	string

Feedback

Arguments

`-gpd` *gpd_dir*

The GPD generated from the StarRC extraction. The argument is the GPD directory.

---

gui_show_source_file

Show the contents of a file in the source browser view.

Syntax

```
status gui_show_source_file
[ -window string ]
-filename string
[ -line int ]
[ -reuse ]
```

Arguments

`-window` *string*

Window specifies the GUI window to show the view in.

This option is optional, the current window is used by default.

`-filename` *string*

Filename specifies the name of the file to be show.

`-line` *int*

Line specifies the line in the file to move to.

`-reuse`

Specifies that an existing Source Browser view can be used instead of creating a new one.

Description

The gui_show_source_file command shows the contents of the specified file in a source browser view. A line can also be specified to automatically move to that line in the file.

If the format of the file is recognised then syntax highlighting will be added and folding of blocks allowed.

Currently only tcl, Verilog and VHDL files are supported and are recognised by their file suffix (.tcl, .v, .vhd/.vhdl).

Feedback

Examples

The following example shows the file "test.tcl" starting at line 100.

```
shell> gui_show_source_file -filename test.txt -line 100
```

See Also

- gui_show_file_in_editor
- gui_show_url_in_browser

---

gui_show_task_assistant

Show the task assistant.

Syntax

gui_show_task_assistant

```
-task          TaskName
-item          ItemName

TaskName         String
ItemName         String
```

Arguments

`-task` *TaskName*

TaskName specifies the name of the task to show in the task assistant.

`-item` *ItemName*

ItemName specifies the name of the task item to show in the task assistant.

Description

The gui_show_task_assistant command opens the task assistant window and displays the page for given task and task item.

See Also

- gui_create_task
- gui_set_current_task
- gui_get_current_task_item
- gui_get_current_task_page

Feedback

gui_show_timing_paths

Shows the details of a collection of timing paths in a timing path inspector.

Syntax

gui_show_timing_paths

```
-paths paths
[-new]
```

Arguments

`-paths`

Collection of timing pathss to be inspected.

`-new`

Creates a new timing path inspector window.

Description

The command shows the details of a collection of timing paths in a timing path inspector. Unless requested by the user, this command will use the most recently used timing path inspector window if it exists. Otherwise it will create a new timing path inspector window.

Examples

The following example inspects a collection of timing paths.

```
prompt> gui_show_timing_paths -paths $paths
```

See Also

- change_selection

gui_show_toolbar

Displays the specified toolbar or all the toolbars in the specified window or for a window type.

Syntax

void *gui_show_toolbar*

```
-toolbar tool_bar_name | -all
[-window window_id]
[-window_type window_type]
```

Feedback

Data Types

```
tool_bar_name      string
window_id          string
window_type        string list
```

Arguments

`-toolbar tool_bar_name`

Specifies the toolbar you want to display. The *-toolbar* option and the *-all* option are mutually exclusive.

`-all`

Displays all the toolbars in the specified window. The *-all* option and the *-toolbar* option are mutually exclusive.

`-window window_id`

Specifies the window in which you want to display the specified toolbar or all toolbars. This option is mutually exclusive with the *-window_type* option. If both *-window* and *-window_type* options are omitted, then by default the tool displays the toolbar or toolbars in the active window.

`-window_type window_type`

Specifies the window types in which you want to display the specified toolbar or all toolbars. All existing windows of the given types will be affected. Toolbar visibility may be set on a window type before a window of the type exists. This option is mutually exclusive with the *-window* option. If both *-window* and *-window_type* options are omitted, then by default the tool displays the toolbar or toolbars in the active window.

Description

This command displays either all the toolbars or the toolbar with the specified name in the window with the specified window ID, or in all windows of the given window type.

Toolbar visibility set by this command will override any visibility setting that has been saved and restored from previous sessions. The last toolbar visibility set by this command or by *gui_hide_toolbar* will be saved and restored on next invocation of the tool if save and restore has not been disabled.

Examples

The following example displays the File toolbar in the active window:

```
prompt> gui_show_toolbar -toolbar File
```

Feedback

The following example displays all the toolbars in the layout window named LayoutWindow.1:

```
prompt> gui_show_toolbar -all -window LayoutWindow.1
```

The following example displays the File toolbar in all TimingWindow type windows:

```
prompt> gui_show_toolbar -all -window_type TimingWindow
```

The following example display all the toolbars in all window types:

```
prompt> gui_show_toolbar -all -window_type [gui_get_window_types -type
 toplevel]
```

See Also

- gui_create_toolbar
- gui_delete_toolbar
- gui_create_toolbar_item
- gui_delete_toolbar_item
- gui_hide_toolbar
- gui_get_toolbar_names
- gui_get_window_types

---

gui_show_url_in_browser

Show the contents of a URL in an external web browser.

Syntax

gui_show_url_in_browser

`-url` *URL*

`URL` *String*

Arguments

`-url` *Url*

Url specifies the url to be displayed.

Feedback

Description

The gui_show_url_in_browser command an external web browser to display the specified web page. The default web browser is defined by the application but is usually 'firefox' on UNIX platforms.

If the web browser supports external comminication (which 'firefox' does) then if no web browser is running the web browser is started. If the web browser is running then the URL is loaded in the browser usually as a new tab.

The default browser can be set using the 'gui_online_browser' variable but the set of valid browsers is usually restricted by the application so not all browsers may be supported. If a unsupported browser is specified the command will fail.

See Also

- gui_online_browser
- gui_show_file_in_editor

---

gui_show_utable

Show the current loaded user tables in a UserTable view.

Syntax

string *gui_show_utable*

```
[-name            Table_Name]
[-window          Window_Name]
[-filter          Filter_Expr]

[-distribution]   [-columns Column_List]
[-report]         [-columns Column]
[-compress]       [-columns Column]

[-plot            Plot_Type]        [-columns Column]

[-import          CSV_Filename      [-tsv_mode] [-ssv_mode] [-report_type]]
[-meta_obj        MetaObj_Name]
[-open            utable_Filename   [-read_only]]

[-mru]
[-top_window]

Table_Name       String
Filter_Expr      String
Column           String
Column_List      StringList
View_Name        String
CSV_Filename     String
```

Feedback

```
MetaObj_Name     String
utable_Filename  String
Plot_Type        box|distribution|scatter
```

Arguments

`-name` *Table_Name*

The name of the user table to show inside the view. If not given, the view will show the default help page. The user can then select a table via the table selector located at the top of the view.

`-window` *Window_Name*

The name of an existing UserTable window/view to activate.

`-filter` *Filter_Expr*

Apply the given filter expression to the opened table view. *Note:* This option is exclusive with the *-import* option. To use this command and filter option you must first use *gui_import_utable*.

`-distribution`

Show table in distribution mode for the column given by the '-columns' option.

`-report`

Show table in report mode for the list of columns given by the '-columns' option.

`-compress`

Show table with column given by '-columns' option, compressed by common bus notations. *Note:* If given a list of columns only the first column is used to compress bus names. Additional columns are given a postfix to indicate the reduction function used on those numeric columns, by default the function is 'MAX', other allowed functions are 'MIN', 'SUM' and 'MEAN'. See example below.

`-plot` *Plot_Type*

Show a plot of the table data for the columns given by the '-columns' option.

`-columns` *Column_List*

Provides a list of one or more column names for the previous options.

`-import` *CSV_Filename*

The name of a CSV (comma seperated values) file to import and create a UserTable from. A column configuration form is presented to allow the user to adjust the final produced UserTable.

Feedback

`-tsv_mode`

This flag changes the default delimiter from the comma char to the tab char.

`-ssv_mode`

This flag changes the default delimiter from the comma char to the semicolon char.

`-report_type`

This flag causes the reader to interpret the file as a report file and convert it into a user table. *Note:* Currently only the 'report_constraint -all_violators' report can be converted.

`-meta_obj` *`MetaObj_Name`*

Optional MetaData object name used to define table meta data. See the command gui_set_utable_meta for more information.

`-open` *`utable_Filename`*

The name of a UserTable native filename (.utable extension) to load into memory and present to the user. *Note*, this option is exclusive with the *-import* option.

`-read_only`

This flag can only be used with the *-open* option. It opens the file in a streaming read only manner, not copying all the data into memory which can save a lot of system memory if the file is huge.

`-mru`

This flag will show the given table in the most recently used User Table view (mru). This option can not be used with the *-window* option.

`-top_window`

This flag will also open a new top level window to house the User Table view in.

Description

```
This command creates a new view (or reuse an existing view) to display
 and
interact with your User Tables. For convenience, options are given for
 you
to also import or open a new table from a file and display it.

User Tables provide a way for you to define and interact with your own
 tables of data.
The data can persist on disk separate from the design database, and is
 completely
under your control. User Tables can be generated from the attributes of a
 set of
```

Feedback

```
selected objects or they can come from external sources via an ASCII
 value file
in CSV/TSV/SSV format.

Interactively these tables can be viewed and support sorting, filtering,
 summary reporting,
value distributions, editing, etc. as well as the ability to
 select/highlight objects in
the design using the object names in columns identified as object type
 columns.

This command is available only when the GUI is running.
```

Examples

The following example just creates and opens a new UserTable view.

```
shell> gui_show_utable
```

```
The following example opens a new UserTable view and displays the
 contents of my_table.
```

```
shell> gui_show_utable -name my_table
```

```
The following example opens a new UserTable view and shows the import
 form used to configure
the table being created from the values file results.csv. If no header
 line is supplied
in the values file the configuration form will use default names that can
 be edited by the
user before committing to table creation.
```

```
shell> gui_show_utable -import results.csv
```

```
The following example opens a new UserTable view and connects to the
 UserTable file
my_table.utable in a read only streaming mode for display.
```

```
shell> gui_show_utable -open my_table.utable -read_only
```

```
The following example used the most recently used UserTable view and
 shows a compressed table
on the column {Cell Name} and assignes the 'SUM' reduction function on
 the given columns.
```

```
shell> gui_show_utable -name my_table -mru -columns { {Cell Name} {Int
 Power:SUM} {Tot Power:SUM} } -compress
```

See Also

- gui_append_utable
- gui_change_selection_utable

Feedback

- gui_close_utable
- gui_create_utable
- gui_export_utable
- gui_get_utable
- gui_import_utable
- gui_set_utable_meta
- gui_open_utable
- gui_write_utable

---

gui_show_window

Show the window in the specified window state

Syntax

```
status gui_show_window
-window window_id
[ -show_state window_state ]
[ { -rect rect | -size isize } ]
```

Data Types

```
window_id       string
window_state    string
rect            {{x1 y1} {x2 y2}}
isize           {width height}
```

Arguments

`-window window_id`

Specifies the toplevel window or view id to show.

`-show_state show_window_state`

Specifies the window state to show the toplevel window or view.

For a toplevel window we have one of:

```
normal     - show at preferred size
maximized  - show maximized (fill whole screen)
minimized  - show iconized
hidden     - hide but do not destroy
```

Note: The window manager may or may not fully honor the specified state. See your window manager documentation for more details.

Feedback

For a view window we have one of:

```
normal    - show at preferred size in view area and raise to top
maximized - show maximized in view area and raise to top
minimized - show iconized in view area
hidden    - hide but do not destroy
```

Note: If displaying maximized then any existing view windows are also maximized. If displaying minimized or normal then any maximized windows are shown as normal.

The default is *SHOW_WINDOW_STATE_NORMAL*.

`-rect rect`

Specifies the size and position of the window.

It is of this format "{{x1 y1} {x2 y2}}", where {x1 y1} specifies the top left location and {x2 y2} specifies the bottom right location of the window.

Note: This option precludes the use of the *-size* option.

`-size isize`

Specifies the width and height of the window.

It is of this format "{width height}"

Note: This option precludes the use of the *-rect* option.

Description

This command shows the specified toplevel window or view in the specified state and optionally sets the geometry.

If no state is specified then it defaults to normal.

Examples

The following examples will create a layout window then use this command to illustrate the various options

```
shell> set top [gui_create_window -type TopLevel]
shell> set layout [gui_create_window -type Layout -parent $top]
shell> gui_show_window -window $layout -show_state {maximized}
shell> gui_show_window -window $layout -show_state {normal}
```

See Also

- gui_close_window
- gui_exist_window

Feedback

- gui_create_window
- gui_get_window_types
- gui_get_window_ids
- gui_set_active_window
- gui_get_current_window

---

gui_show_window_toolbar

Show toolbar in view

Syntax

```
status gui_show_window_toolbar
-window window_id
[ -toolbar string ]
```

Data Types

```
window_id  string
```

Arguments

`-window window_id`

Specifies the view name id.

Note: When in an intialization or control callback the window and toolbar will be defaulted to the current values so this option does not need to be specified.

`-toolbar string`

Specifies the optional toolbar name to show.

Note: the command will fail if the set does not exist

Description

This command shows the specified view's toolbar.

Note: a view's toolbar will only be shown if it has controls in it.

Examples

The following example will show a Charts View and show the toolbar.

```
shell> set top [gui_create_window -type TopLevel]
shell> set charts [gui_create_window -type Charts -parent $top]
shell> gui_show_window_toolbar -window $charts
```

Feedback

See Also

- gui_create_window
- gui_hide_window_toolbar

---

gui_start

Starts the application GUI.

Syntax

```
status gui_start
[-file script]
[-no_windows]
[-offscreen 1|0]
[-- x_args ...]

string script
```

Arguments

`-file script`

The given script file is sourced before the GUI starts.

`-no_windows`

The GUI starts without showing the default window.

`-offscreen`

If the specified value is 1 then the GUI starts in offscreen mode.

If the specified value is 0 then the GUI starts in normal (X Display) mode.

`-- x_args`

X11-specific arguments to pass to the X connection.

Description

This command starts the application GUI from the shell prompt. It is ignored if the application GUI has already been started. start_gui is an alias for gui_start.

Note the application can only ever connect to one X server during program execution, so changing the value of the DISPLAY environment variable after the first successful gui_start command has no effect.

Examples

The following example starts the application GUI.

shell> *gui_start*

Feedback

See Also

- gui_stop
- start_gui
- stop_gui

---

gui_stop

Stops the application GUI.

Syntax

```
status gui_stop
 [-close]
```

Arguments

-close

Close the GUI so it can be restarted on a different display.

Without this option (the default) the GUI is just disabled and hidden and can be quickly restored using gui_start.

The limitation of not using close is that the same display must be used when the GUI is started again.

Description

This command stops the application GUI and returns to the shell prompt.

It is ignored if the application GUI has not been started or has been stopped.

stop_gui is an alias for gui_stop.

Examples

The following example stops the application GUI and returns to the shell prompt.

shell> *gui_stop*

See Also

- gui_start
- start_gui
- stop_gui

Feedback

gui_update_attrgroup

Updates a group of attributes for an object type.

Syntax

```
status gui_update_attrgroup
-class design_object
-name name
[-attr_list {{attr_1}...{attr_n}}]
[-add]
[-delete]
[-move up | down | top | bottom | after | before]
[-attr attribute]
[-anchor attribute]
```

Data Types

```
design_object      string
name               string
attr_1             string
attr_n             string
attribute          string
```

Arguments

`-class design_object`

Specifies the type of design object.

`-name name`

Specifies the non-editable name of the attribute group to update for the specified object type.

`-attr_list {{attr_1}...{attr_n}}`

Lists and orders the attributes to be included in the group.

`-add`

Adds the attribute specified by the *-attr* option to the end of attributes list, by default, or after the attribute specified by the *-anchor* option.

`-delete`

Removes the attributes specified by the *-attr* option from the attribute list. Note that the attribute still exists.

`-move up | down | top | bottom | after | before`

Moves the attribute specified by the *-attr* option in the specified direction. If you specify *before* or *after* to move the attribute relative to the position of a reference attribute, use the *-anchor* option to specify the reference attribute.

Feedback

`-attr attribute`

Specifies the attribute to be added, removed, or moved with the *-add*, *-delete*, or *-move* option.

`-anchor attribute`

Specifies the reference attribute to be used with the *-add* or *-move* option.

Description

The *gui_update_attrgroup* command updates an attribute group for the specified object type. You can reset the entire attribute list by using the *-attr_list* option, or you can change the list by adding, removing, or moving with a single attribute.

The order of the attributes in an attribute group is important and is set by the *-attr_list* option when you create or update the group.

Examples

```
prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \\
   -attr_list { "TestAttr1" "TestAttr2" "TestAttr3" }

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \\
   -add -attr "TestAttr5"

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \\
   -add -attr "TestAttr5" -anchor "TestAttr2"

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \\
   -delete -attr "TestAttr5"

prompt> gui_update_attrgroup -class Cell -name "TestGroup2" \\
   -move down -attr "TestAttr2"
```

See Also

- gui_create_attrgroup
- gui_delete_attrgroup
- gui_list_attrgroups

---

gui_update_pref_file

Save current application preferences to the user preference file.

Syntax

string *gui_update_pref_file* [-file *FilePathName*]

`FilePathName` *String*

Feedback

Arguments

`-file FilePathName`

FilePathName specifies the full pathname of the user preference file to write the current preferences to. If this option is missing, the file to write to is controlled by the gui variables : pref_file_name and pref_file_path. These two variables are set with the gui_set_var command and their values retrieved with the gui_get_var command.

Description

This command updates the user preference file. Normally, application will automatically save user preferences to a default system preference file on exit, and retrieve the preferences from the default user preference file on application start. So, this command is rarely used, except internally.

See Also

- gui_create_pref_category
- gui_create_pref_key
- gui_set_pref_value
- gui_get_pref_value
- gui_get_pref_value_type
- gui_remove_pref_key
- gui_get_pref_categories
- gui_get_pref_keys
- gui_exist_pref_category
- gui_exist_pref_key

---

gui_update_vm

Update Visual Mode

Syntax

string *gui_update_vm* -name *identifier*

`string identifier`

Feedback

Arguments

`-name identifier`

Specifies the visual mode name whose associated tcl update command will be executed. The tcl update command is associated with the visual mode using the -update_cmd option to the gui_create_vm or gui_set_vm commands.

Description

Execute a visual mode's associated tcl update command.

Examples

To force the visual mode "vm1" to execute its associated update command, use the following:

shell> *gui_update_vm -name vm1*

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_get_vm
- gui_get_vmbucket
- gui_list_vm
- gui_remove_vm
- gui_remove_vmbucket
- gui_set_vmbucket
- gui_set_vm

---

gui_update_vm_annotations

Updates visual mode annotations.

Syntax

string *gui_update_vm_annotations*

-clear -center -add *points* -draw_net *style* [-type *shape*] [-text *message*] [-color *color*] [-pattern *pattern*] [-line_style *line_style*] [-width *width*] [-info_tip *info_tip*] [-query_text *query_text*] [-query_command *query_command*] *object*

Feedback

Data Types

points	list
style	string
shape	string
message	string
color	string
pattern	string
line_style	string
width	integer
info_tip	string
query_text	string
query_command	string
object	string

Arguments

`-clear`

Removes all annotations from the specified object.

`-center`

Interprets all object locations as the centers of the objects rather than their origins.

`-add` *points*

Adds points for the annotation shape that you specify with the *-type* option. The points are specified by a tool command language (Tcl) list in which each element is one of the following:

- the {x y} location of a point
- a collection containing a single object that has a visual representation in the layout. In other words, it is not a net, for example. The location of the point is the origin of the object.

If a collection is specified the list element can optionally include a position type, one of:

- center : the annotation is placed at the center of the largest rectangle of the object. The largest rectangle is the rectangle with the most area which is completely contained inside the object shape.
- bbox_center : the annotation is placed at the center of the bounding box of the object.
- bbox_ll : the annotation is placed at the lower left of the bounding box of the object.
- bbox_lr : the annotation is placed at the lower right of the bounding box of the object.

- bbox_ul : the annotation is placed at the upper left of the

  bounding box of the object. • bbox_ur : the annotation is placed at the upper right of the bounding box of the object.

If a bounding box position type is specified, i.e. one of 'bbox_center', 'bbox_ll', 'bbox_lr', 'bbox_ul' or 'bbox_ur', then the list element can also optionally include an x and y fractional offset from this bounding box point. The x and y offset is a fraction of the width and height of the object's bounding box respectively. For example, for a 'bbox_center' position type, an x offset of -0.5 is the left edge and an x offset of 0.5 is the right edge. Similarly a y offset of -0.5 is the bottom edge and a y offset of 0.5 is the top edge.

For annotations referring to a single-object collection, if the object is moved, the tool updates the point automatically as needed.

`-draw_net` *`style`*

Specifies how the nets are displayed in the layout view for this annotation if the specified objects are nets or physical buses.

This annotation overrides the current net display setting on the View Settings panel. The valid *style* values are

flyline
routing
routing_flyline

`-type` *`shape`*

Specifies the type of annotation. The valid *shape* values are

rect
line
arrow
polygon
polyline
text

`-text` *`message`*

Specifies a text string that you want to display.

`-color` *`color`*

Specifies the color. The default is the bucket color.

`-pattern` *`pattern`*

Specifies the fill pattern. The default is none.

Feedback

`-line_style` *line_style>*

Specifies the line style. The default is none.

`-width` *width*

Specifies the line width. The default is 1.

`-info_tip` *info_tip*

Specifies an info tip for this annotation. When the user starts the query tool in the layout, and puts the mouse cursor over this annotation the specified text will be displayed as the infoTip.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

```
%window  the layout window name
%view    the layout view name
%object  a collection containing the annotation
```

`-query_text` *query_text*

This option is only valid if an info tip was specified. Use this to specify a more detailed string that is displayed in the query palette when the object is selected via the query tool. If query text is not specified then the query tool will just use the info tip string.

If the text starts with a '=' character the text after it will be considered to be a tcl command which, when executed, will return the query text to be displayed. The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

```
%window  the layout window name
%view    the layout view name
%object  a collection containing the annotation
```

`-query_command` *query_command*

This option is only valid if an info tip was specified. Use this to specify a tcl command to be run when the user clicks on the annotation.

The tcl command can contain special strings, which are replaced with details of the layout window and annotation, so this information can be used in the tcl procedure.

```
%window  the layout window name
%view    the layout view name
```

Feedback

```
%object  a collection containing the annotation
%x       the x position clicked in design coordinates
%y       the y position clicked in design coordinates
```

object

Specifies the object with which to associate the annotation.

Description

This command adds annotations to objects returned by the *gui_create_vm_objects* command.

Layout view visual modes are used to color objects in the layout view based on certain criteria. You can use the *gui_update_vm_annotations* command to display not only object coloring but some additional annotations for the objects in a visual mode bucket.

Examples

To create a bucket that displays cells and a line connecting those cells to the location (0,0), enter code like the following example:

```
prompt> set objs [gui_create_vm_objects $cells]
   foreach_in_collection o $objs {
      set c [get_attribute $o core_obj]
      gui_update_vm_annotations $o -type line -add [list {0 0} $c]
   }
prompt> gui_create_vmbucket -vmname $vm -name $bucket -color red
 -collection $objs
```

See Also

- gui_create_vm
- gui_create_vmbucket
- gui_create_vm_objects

---

gui_view_port_history

Does the gui_view_port_history operations.

Syntax

gui_view_port_history

```
[-window window_name]
    [-next]
    [-previous]
    [-add name]
    [-to  name]
```

Feedback

```
[-list_names]
[-rect bounding box]
[-delete_name]
[-isprevious]
[-isnext]
[-dialog]
[-tcl_list]
[-help_cmd]
```

Arguments

`-window fwindow_name`

Give the name of a window. this is required option except if you use *-dialog* option.

`-next`

Specify if you want to go to previous stored view port history. If you use *-next* option you can not use any other option except *-window* option: they are mutually exclusive.

`-previous`

Specify if you want to go to previous stored view port history. If you use *-previous* option you can not use any other option except *-window* option: they are mutually exclusive.

`-add name`

Current view port will be stored by this name. *-add* option can be combined with *-rect* option. In that case the view port history with the given rect instead of current view port will be saved in to view port history with the given name.

If you use *-add* option you can not use any other option except *-window* and *-rect* option : they are mutually exclusive.

`-to name`

Give the name of a view port history already stored. If you use *-to* option you can not use any other option except *-window* option: they are mutually exclusive.

`-list_names`

Specify if you want to see list of names of all the named view port history. If you use *-list_names* option you can not use any other option except *-window* option: they are mutually exclusive.

`-delete_name name`

The already stored view port history with this name will be deleted. If you use *-delete_name* option you can not use any other option except *-window* option: they are mutually exclusive.

Feedback

`-rect` *`boundingbox`*

Adds give the bounding box (in design units) to unnamed view port history. The values should be given in following format. format: {{x1 y1} {x2 y2}} *-rect* option can be combined with *-add* option. In that case the view port history with the given rect instead of current view port will be saved in to view port history with the given name. If you use *-add* option you can not use any other option except *-window* and *-add* option : they are mutually exclusive.

`-isnext`

return true returns true if there previous zoom history. If you use *-isnext* option you can not use any other option except *-window* option: they are mutually exclusive.

`-isprevious`

returns true if there next zoom history. If you use *-isprevious* option you can not use any other option except *-window* option: they are mutually exclusive.

`-dialog`

specify to invoke dialog for named view port history. If you use *-dialog* option you can not use any other option except *-help_cmd* option: they are mutually exclusive.

`-tcl_list`

return a tcl_list of named view port history If you use *-tcl_list* option you can not use any other option except *-window* option: they are mutually exclusive.

`help_cmd` *`help_command`*

Give a command which will be executed when help buttonis clicked in the dialog.

Description

This command store view port for the application. At least one option needs to be specified.

last 100 View ports will be stored in view port history, which can be accessed by specifying *-previous* and *-next* options.

When you use *-add* then the current view port will be stored by the given name in the to view port history.

when you use *-to* then the view port stored by this name will be shown and added to the view port history.

Examples

The following example zooms in to the given rectangle.

Feedback

```
fpc_shell> gui_view_port_history -rect {{-2232.321 -536.887} {141.286
 2175.807}}
```

The following example zooms to next .

```
fpc_shell> gui_view_port_history -next
```

The following example zooms to previous.

```
fpc_shell> gui_view_port_history -previous
```

The following example gives list of all the named zooms.

```
fpc_shell> gui_view_port_history -list_names
```

The following example puts the current zoom in to zoom history by given name.

```
fpc_shell> gui_view_port_history -add xyz
```

The following example zooms to already stored to zoom .

```
fpc_shell> gui_view_port_history -to xyz
```

The following example deletes an already stored to zoom.

```
fpc_shell> gui_view_port_history -delete xyz
```

---

gui_write_annotations

Saves annotations to a Tcl file.

Syntax

```
status gui_write_annotations
-file filename
[-append]
[-force]
[-compress method]
[annotations]
```

Data Types

```
filename      string
method        string
annotations   collection
```

Arguments

`-file filename`

Specifies the name of the file in which the tool saves the annotation script.

Feedback

`-append`

Append the annotation script to an existing file instead of creating a new one. The specified file must exist. This option cannot work with *-force*.

`-force`

Overwrite the specified file, if it exists, with the new one. This option cannot work with *-append*.

`-compress` *method*

Specifies the compression type to use when writing the file. By default, no compression is performed.

`annotations`

Specifies a collection of annotations to write. Get a collection of annotations with the gui_get_annotations command. If this option is not specified, then all the annotations with *global* window handle and *global* group type would be written by, annotations with specific window handle or group type are excluded.

Description

This command writes a collection of annotations to a Tcl script. The attribute -visible will only be exported if it is set as invisible.

Examples

```
Write out all global annotations.

prompt> gui_write_annotations -file my_annotations.tcl

Write out all annotations in Layout.1.

prompt> gui_write_annotations -file my_annotations.tcl \\
    [gui_get_annotations -window Layout.1]
```

See Also

- gui_add_annotation
- gui_get_annotations
- gui_remove_all_annotations
- gui_remove_annotations

---

gui_write_charts_data

Write charts data

Feedback

Syntax

```
status gui_write_charts_data
-view view_name
[ -file file_name ]
```

Data Types

```
view_name  string
file_name  string
```

Arguments

`-view view_name`

View to output data for.

`-file file_name`

Filename to output data to. If not specified then the the output is sent to the terminal.

Description

Write charts data as a tcl script to terminal or a file.

This command allows the user to save the state of the plots in the current view to a tcl command which can be source'd to reproduce the plot.

Details of the models loaded for the view's plots, the views annotations, the plots and the plots annotations are saved.

Only the view and plot properties that have changed from their default values are saved.

Note: Only the filename, tcl variable or collection name of the model data can be saved so these external dependencies, file data, contents of the tcl variable, or contents of the collection will need to be recreated before the script can be used.

Examples

The following example writes view data to the file "view_data.tcl".

```
shell> gui_write_charts_data -view Charts.1 -file "view_data.tcl"
```

See Also

- gui_create_charts_model

---

gui_write_charts_image

Save charts plot or view to image file

Feedback

Syntax

```
status gui_write_charts_image
-view view_name
-plot plot_name
-file string
```

Data Types

```
view_name  string
plot_name  string
```

Arguments

`-view view_name`

Charts view to save (all charts).

`-plot plot_name`

Single charts plot in view to save.

`-file string`

Output filename.

The file suffix is used to determine type i.e. .png for PNG file or .svg for SVG file.

If the file suffix (converted to lower case) is not .png or .svg then PNG is used.

Description

Save charts plot or view to PNG or SVG image file.

Examples

The following example saves all charts in a view to a PNG file charts.png.

```
shell> gui_write_charts_image -view $view -file charts.png
```

---

gui_write_timing_paths

Writes a persistent path data file containing a timing path collection.

Syntax

```
status gui_write_timing_paths
-file name
[-overwrite]
[-design name]
[-lib name]
[-view name]
[-label name]
[-scenario name]
```

Feedback

```
[-mode name]
[-corner name]
[-tag name]
[-comment name]
clct_name
```

Data Types

```
name       string
clct_name  collection
```

Arguments

`-file name`

Specifies the name of the persistent path data file to be written. By default, if the file already exists, the command fails.

`-overwrite`

Specifies the file that already exists will be overwritten.

`-design name`

Set (overwrite) the design name for the given collection.

`-lib name`

Set library name for the given collection.

`-view name`

Set view name [default name = 'design']

`-label name`

Set label name.

`-scenario name`

Set scenario name [default name = 'default']

`-mode name`

Set mode name [default name = 'default']

`-corner name`

Set corner name [default name = 'default']

`-tag name`

User can tag timing path file with some arbitrary string.

`-comment name`

User comment for timing path file.

clct_name

Specifies the collection of timing paths to be written.

Description

The *gui_write_timing_paths* command writes a persistent path data file containing the given path collection. The persistent path data file contains all the information needed to recreate those timing paths using the *gui_read_timing_paths* command in ICC2.

For correct import to ICC2, you must provide a legal 'library', 'view', 'scenario', 'mode' and 'corner'.

The user also has the option of adding a custom tag and comment to the file that can be quickly read by ICC2 without having to load all the timing paths.

Examples

The following examples shows how a timing path collection can be written to a persistent path data file. Here we allow the write to use the current design name, and use 'default' for the scenario, mode and corner name.

```
> gui_write_timing_paths $paths -file "my.paths" -lib name -view name
```

---

gui_write_user_map

write snapshot of current map mode data to file.

Syntax

string *gui_write_user_map*

```
-name mapName
-file outputFile
[-metadata]       (Export header only)
```

Data Types

mapName string

outputFile string

Arguments

-map *mapName*

Specifies the current Visual/Map Mode name in layout

-file outputFile

Specifies the name of output file. It can be a single file name(generated in previous working directory) or have a absolute path.

`-metadata`

This option indicates that a meta data section should be included in the output. The meta data contains basic information of the map.

For displaying the output data as a user-map, including the meta data is recommanded.

For processing the data without displaying as user-map or other purposes, this option should not be turned on.

Description

Write snapshot of current map mode data into csv file. The map data is easier to read and adapt to other app. User map is a general feature and will not have all of the features of the map that was written only the basic data presentation and bucket settings.

Here's an example of the output file of cell density map:

```
#META_DATA
#type,name,property,value
#table,,tag,gui_write_user_map
#table,,app_name,icc2
#table,,version,"V1.0"
#table,,report_date,"2029-Jul-29 02:55:51"
#table,,title,"Cell Density"
#table,,bins,10
#table,,min_threshold,0.100000
#table,,max_threshold,1.100000
#table,,color_scale,TemperatureScale
#column,Coordinates,type,poly_rect
#column,value,type,double
#column,Comment,type,string
#END_META_DATA

Coordinates,Value,Comment
"{12.600 12.600} {25.200 0.000}",0.161270,0.16127
"{25.200 12.600} {37.800 0.000}",0.551746,0.55175
"{37.800 12.600} {50.400 0.000}",0.643175,0.64317
"{50.400 12.600} {63.000 0.000}",0.613333,0.61333
"{63.000 12.600} {75.600 0.000}",0.579048,0.57905
```

Exanmple of Global Route Congestion map:

```
#META_DATA
#type,name,property,value
#table,,tag,gui_write_user_map
#table,,app_name,icc2
#table,,version,"V1.0"
#table,,report_date,"1989-Jun-04 02:30:28"
#table,,title,"Global Route Congestion"
#table,,bins,9
#table,,min_threshold,0.000000
```

```
#table,,max_threshold,7.000000
#table,,color_scale,TemperatureScale
#column,Coordinates,type,poly_rect
#column,value,type,int
#column,Comment,type,string
#column,Special,type,string
#END_META_DATA

Coordinates,Value,Comment,Special
"{0.000 0.000} {2.520 0.000}",-23,-23/23
"{0.000 2.520} {0.000 0.000}",-26,-26/26
"{2.520 0.000} {5.040 0.000}",-26,-26/26
"{2.520 2.520} {2.520 0.000}",-26,-26/26
"{5.040 0.000} {7.560 0.000}",-24,-24/25
"{5.040 2.520} {5.040 0.000}",-25,-25/26
"{7.560 0.000} {10.080 0.000}",-26,-26/26
```

Examples

While the map is generated and presented in layout:

```
prompt>gui_wirte_user_map -map cellDensityMap -file cellden.csv
```

See Also

- gui_read_user_map
- gui_remove_user_map

---

gui_write_utable

Write the UserTable to file in native UserTable format.

Syntax

string *gui_write_utable*

```
-name         Table_Name
[-file        File_Name]
[-filter      Filter]
[-read_only]
[-overwrite]

Table_Name    String
File_Name     String
```

Arguments

`-name` *Table_Name*

The name of the user table to write out to a file.

Feedback

`-file File_Name`

The name of the file to write the table to, by default the name of the table is used with the '.utable' extension.

`-filter Filter`

This argument allows you to filter the rows in the given table to only rows that match the filter. The filter expression is the same expression allowed in the GUI filter field of the UserTable GUI view.

`-read_only`

This flag closes the table that has been written out and then re-opens it in read-only (streaming) mode which can save a lot of system memory if the table is huge.

`-overwrite`

This flag will force the destination file to be overwritten.

Description

```
This command writes the given table name to a file in the native user
 table format,
preserving all meta data (filter, sort order and column information). The
 UserTable
file can then be later re-opened to copy the information back into memory
 or opened
in a read-only (streaming) mode using the command gui_open_utable.
```

Examples

```
The following example writes out the given table to a file name equal to
 the table name
plus the extension '.utable'.

shell> gui_write_utable -name my_table

The following example writes out the given table (overwrites the file if
 necessary) and
then closes the table in memory and re-opens it in read-only (streaming)
 mode.

shell> gui_write_utable -name my_table -read_only -overwrite
```

See Also

- gui_append_utable
- gui_change_selection_utable
- gui_close_utable

Feedback

- gui_create_utable
- gui_export_utable
- gui_import_utable
- gui_open_utable
- gui_show_utable

---

gui_write_window_image

Saves an image of a view or toplevel window in the specified file

Syntax

```
status gui_write_window_image
[ -file filename ]
[ -format image_format ]
[ -window window_name ]
[ -palette palette_type ]
[ -size window_size ]
[ -clip ]
```

Data Types

```
filename       string
image_format   string
window_name    string
palette_type   string
window_size    {width height}
```

Arguments

`-file filename`

Specifies the name of the file in which the tool saves the window image.

If you include a file name extension, it determines the image format unless you also specify the *-format* option.

If no file is specified it defaults to 'gui_image.png'.

`-format png | xpm | jpg | bmp`

Specifies the image format to be used in the file. The default value is png.

Note that if the format that you specify with the *-format* option is different from the format specified by the file name extension, the *-format* value takes precedence and the tool appends an additional extension to the file name.

Feedback

`-window` *`window_name`*

Specifies the name of the window for which you want to save an image in the specified image file.

You can view a list of the window names for all the open toplevel and view windows by using the *gui_get_window_ids* command.

By default, the tool saves an image of the most recently active view window.

`-palette` *`palette_type`*

Specifies the type or side of the palette for which you want to save an image in the specified image file. When a type is specified just the palette of that type is saved, if a side is specified all palettes on that side are saved.

If *-window* option is specified the palettes from that window are used otherwise palettes from the most current window are used.

Note: The palette type must exist and be visible. You can use the command *gui_show_palette* to display the palette.

You can view a list of the palette types by using the *gui_get_palette_types* command. The side can be 'left, right, top or bottom'.

`-size` *`window_size`*

The size of the image to be generated. If not specified then the full window size is used.

`-clip`

Clip the image to the contents skipping horizontal or vertical empty areas due to diffences in the aspect ratio of the view and data being displayed.

Only currently supported for Layout and Charts view windows.

Description

The *gui_write_window_image* command takes a snapshot of a specified toplevel window or child view window in a specified image format and writes the result to a specified file.

Examples

The following example writes a snapshot of the TopLevel.1 window to a image file "snapshot.png".

```
shell> gui_write_window_image -window TopLevel.1 -file snapshot.png
```

The following example writes a snapshot of the Layout.1 view to a image file "layout.png" of size 800 pixels by 600 pixels.

Feedback

```
shell> gui_write_window_image -window Layout.1 -file layout.png -size
 {800 600}
```

The following example writes a snapshot of the all palettes on the right side of the current active window to an image file "palettes.png".

```
shell> gui_write_window_image -palette right -file palettes.png
```

The following example writes a clipped snapshot of the current layout view to an image file "layout.png".

```
shell> set window [gui_get_current_window -mru -type Layout]
shell> gui_write_window_image -window $window -clip -file layout.png
```

See Also

- gui_get_window_ids
- gui_get_current_window
- gui_show_palette

---

gui_zoom

Change the viewport of a view

Syntax

gui_zoom

-window *window* [-fit | -exact] [-full] [-selection] [-rect *{{lx ly} {ux uy}}*] [-factor *factor*] [-at_point *{x y}*] [-clct *clct*] [-zoom_in_mode] [-zoom_out_mode] [-select_mode]

```
window          String
rect            String
factor          Float
```

Arguments

`-window` *window*

window specifies the window of which this command should change the viewport.

`-full`

Zoom such that all objects are visible.

`-factor` *`factor`*

Keep the center of the viewport and zoom by factor *factor*. A *factor*" > 1 causes a zoom-in, a *factor*" < 1 causes a zoom-out. The argument must be greater than zero.

`-at_point` *`{x y}`*

Can only be specified with -factor argument. If supplied, zoom is performed at specified point instead of viewport center. The option is not supported by all window types.

`-fit`

Effects the -rect, -selection and -clct arguments. If supplied, will cause the zoom to not over zoom. Will zoom a reasonable distance away from the rectangle or objects being zoomed. The definition of "reasonable" is window dependent. This argument is mutually exclusive with -exact.

`-exact`

Effects the -rect, -selection and -clct arguments. If supplied, will cause the zoom to zoom exactly to the arguments. This argument is mutually exclusive with -fit.

`-selection`

Zoom such that all selected objects that are represented in the window are visible and if possible are easy to see and are shown in a reasonable context. If neither -fit or -exact is supplied, -fit is assumed.

`-rect` *`{{lx ly} {ux uy}}`*

Zoom such that the window shows the rectangle *rect* in its viewport. What coordinates are assumed for *rect* are dependent on the window. If neither -fit or -exact is supplied, -exact is assumed.

`-clct` *`clct`*

Zoom such that all objects in *clct* that are represented in the window are visible and if possible, are easy to see and are shown in a reasonable context. If neither -fit or -exact is supplied, -fit is assumed.

`-zoom_in_mode`

Put the window in zoom-in mode.

`-zoom_out_mode`

Put the window in zoom-out mode.

`-select_mode`

Put the window in selection mode.

Feedback

Description

The *gui_zoom* controls the viewport of a view that supports zooming. Not all views need to support all of the functionality provided by the interface of this command.

Examples

The following example has the layout view show the entire design and then zooms in by a factor of 2, i.e. only a quarter of the area of the design is still visible:

```
gui_zoom -window Layout.1 -full
gui_zoom -window Layout.1 -factor 2
gui_zoom -window Layout.1 -rect {{0.0 0.0} {123.456 500.123}}
gui_zoom -window Layout.1 -rect {{0.0 0.0} {123.456 500.123}} -fit
gui_zoom -window Layout.1 -selection -exact
```

---

gui_zoom_all_layouts_to_current_view

Rescales all layout views to the current view.

Syntax

gui_zoom_all_layouts_to_current_view

Arguments

None.

Description

The command sets the zoom factor of all layout views to the zoom factor of current layout view.

Examples

```
prompt> gui_zoom_all_layouts_to_current_view
```

See Also

- gui_zoom

---

gui_zoom_to_selected_errors

Layout window zoom to errors selected in the error browser.

Syntax

string *gui_zoom_to_selected_errors*

Feedback

Description

If you have selected the errors in the error browser, you can execute this command to have layout zoom to the selected errors.

Examples

If you select errors, then perform layout zooming operations, the following command will force layout to zoom to selected errors: gui_zoom_to_selected_errors

See Also

- gui_error_browser
- gui_get_errors
- gui_set_current_errors
- gui_set_selected_errors
- gui_set_error_browser_option

h

help

Displays quick help for one or more commands.

Syntax

```
string help
[-verbose]
[-groups]
[pattern]
```

Data Types

pattern	string

Arguments

`-verbose`

Displays the command options; for example *command_name -help*.

`-groups`

Displays a list of command groups only.

pattern

Displays commands matching the specified pattern.

Description

The *help* command is used to get quick help for one or more commands or procedures. This is not the same as the *man* command that displays reference manual pages for a command. There are many levels of help.

By typing the *help* command, a brief informational message is printed followed by the available command groups.

List all of the commands in a group by typing the group name as the argument to *help*. Each command is followed by a one-line description of the command.

To get a one-line help description for a single command, type *help* followed by the command name. You can specify a wildcard pattern for the name; for example, all commands containing the string "alias". Use the *-verbose* option to get syntax help for one or more commands. Use the *-groups* option to show only the command groups in the application. This option cannot be combined with any other option.

Examples

The following example lists the commands by command group:

```
prompt> help
Specify a command name or wild card pattern to get help on individual
commands. Use the '-verbose' option to get detailed option help for a
command. Commands also provide help when the '-help' option is passed to
the command.'

You can also specify a command group to get the commands available in
that group. Available groups are:

  Procedures:                Miscellaneous procedures
  Help:                      Help commands
  Builtins:                  Generic Tcl commands
 ...
```

The following example displays the list of procedures in the Procedures group:

```
prompt> help procedures
 ls                 # List files
 sh                 # Execute a shell command
```

The following example uses a wildcard character to display a one-line description of all commands beginning with *a*:

```
prompt> help a*
 alias                # Create a command which expands to words.
```

Feedback

```
 append                # Builtin
 array                 # Builtin
```

This example displays option information for the *source* command:

```
prompt> help -verbose source
 source                # Read a file and execute it as a script
   [-echo]                 (Echo all commands)
   [-verbose]              (Display intermediate results)
   file_name               (Script file to read)
```

See Also

- man
- sh_help_shows_group_overview

---

help_attributes

Display help for attributes and object types

Syntax

string *help_attributes* [-verbose]

[-application] [-user] [*class_name*] [*attr_pattern*]

```
string  class_name
string  attr_pattern
```

Arguments

`-verbose`

Display attribute properties.

`-application`

Only display application defined attributes.

`-user`

Only display user defined attributes.

`class_name`

Object class to retrieve help for.

`attr_pattern`

Show attributes matching pattern.

Feedback

Description

The *help_attributes* command is used to get quick help for the set of available attributes. When this command is run with no arguments a brief informational message is printed followed by the available object classes.

Examples

```
prompt> help
cci_test> help_attributes
Specify an object type and an optional wild card pattern to get
 information
on the available attributes defined for the specified object type. Use
 the
-verbose option to get detailed information on the attributes

The available object types are:

  cell        net       pin
  ...
```

See Also

- get_attribute
- help
- get_defined_attributes

---

history

Displays or modifies the commands recorded in the history list.

Syntax

string *history*

```
[-h]
[-r]
[argument_list]
```

Data Types

argument_list	list

Arguments

`-h`

Displays the history list without the leading numbers. You can use this for creating scripts from existing history. You can then source the script with the

source command. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

`-r`

Reverses the order of output so that most recent history entries display first rather than the oldest entries first. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

`argument_list`

Additional arguments to *history* (see DESCRIPTION).

Description

The *history* command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." The most commonly used forms of the command are described below. You can combine each with either the *-h* or *-r* option, but not both.

- With no arguments, the *history* command returns a formatted string (intended for you to read) giving the event number and contents for each of the events in the history list.
- If a single, integer argument *count* is specified, only the most recent *count* events are returned. Note that this option is a nonstandard extension to Tcl.
- Initially, 20 events are retained in the history list. You can change the length of the history list using the following:

  history keep count

Tcl supports many additional forms of the *history* command. See the "Advanced Tcl History" section below.

Examples

The following examples show the basic forms of the *history* command. The first is an example of how to limit the number of events shown using a single numeric argument:

```
prompt> history 3
     7  set base_name "my_file"
     8  set fname [format "%s.db" $base_name]
     9  history 3
```

Using the *-r* option creates the history listing in reverse order:

```
prompt> history -r 3
     9  history -r 3
     8  set fname [format "%s.db" $base_name]
     7  set base_name "my_file"
```

Using the *-h* option removes the leading numbers from each history line:

```
prompt> history -h 3
set base_name "my_file"
set fname [format "%s.db" $base_name]
history -h 3
```

Advanced Tcl History

The *history* command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." When specifying an event to the *history* command, the following forms may be used:

- A number, which if positive, refers to the event with that number (all events are numbered starting at 1). If the number is negative, it selects an event relative to the current event; for example, *-1* refers to the previous event, *-2* to the one before that, and so on. Event *0* refers to the current event.
- A string selects the most recent event that matches the string. An event is considered to match the string either if the string is the same as the first characters of the event, or if the string matches the event in the sense of the *string match* command.

The *history* command can take any of the following forms:

history

Same as *history info*, described below.

history add command [exec]

Adds the *command* argument to the history list as a new event. If *exec* is specified (or abbreviated), the command is also executed and its result is returned. If *exec* is not specified, an empty string is returned.

history change newValue [event_number]

Replaces the value recorded for an event with *newValue*. The *event_number* specifies the event to replace, and defaults to the *current* event (not event *-1*). This command is intended for use in commands that implement new forms of history substitution and want to replace the current event (that invokes the substitution) with the command created through substitution. The return value is an empty string.

history clear

Erases the history list. The current keep limit is retained. The history event numbers are reset.

history event [event_number]

Returns the value of the event given by *event_number*. The default value of *event_number* is *-1*.

history info [count]

Returns a formatted string, giving the event number and contents for each of the events in the history list except the current event. If *count* is specified, then only the most recent *count* events are returned.

history keep [count]

Changes the size of the history list to *count* events. Initially, 20 events are retained in the history list. If *count* is not specified, the current keep limit is returned.

history nextid

Returns the number of the next event to be recorded in the history list. Use this for printing the event number in command-line prompts.

history redo [event_number]

Reruns the command indicated by *event* and returns its result. The default value of *event_number* is *-1*. This command results in history revision. See the following section for details.

History Revision

Pre-8.0 Tcl had a complex history revision mechanism. The current mechanism is more limited, and the *substitute* and *words* history operations have been removed. The *clear* operation was added.

The *redo* history option results in much simpler "history revision." When this option is invoked, the most recent event is modified to eliminate the history command and replace it with the result of the history command. If you want to redo an event without modifying the history, use the *event* operation to retrieve an event, and use the *add* operation to add it to history and execute it.

i

identify_interface_logic

Sets the *is_interface_logic_pin* attribute on pins of the current design that are part of its interface logic.

Feedback

Syntax

status *identify_interface_logic*

```
[-ignore_ports port_list]
[-auto_ignore]
[-latch_level levels]
[-context_borrow]
[-keep_ignored_fanout]
[-include_pins pin_list]
[-critical_pins]
[-include_all_net_pins]
[-traverse_disabled_arcs]
```

Data Types

port_list	list
levels	integer
pin_list	list

Arguments

`-ignore_ports` *port_list*

Specifies a list of input or output ports whose fanout or fanin is to be ignored when placing the *is_interface_logic_pin* attribute. Substitute the list of ports you want for the *port_list* value. Clock ports are automatically ignored; you do not have to specify them in this list.

You can use this option to exclude the fanout of ports connected to chip-level nets (for example, scan enable and reset) from affecting the contents of interface logic. If these nets are not explicitly ignored, they cause unnecessary logic (for example, internal registers) to be part of the interface logic on the current design. You can also use this option to selectively generate the interface logic for a subset of the ports on a block; for example, an interface logic model (ILM) can model only the timing behavior on a subset of the ports on a block.

By default all nonclock input and all output ports are used in identifying the contents of interface logic. The *-ignore_ports* and *-auto_ignore* options are mutually exclusive.

`-auto_ignore`

Enables automatic determination of ports whose fanout should be ignored when placing the *is_interface_logic_pin* attribute. A port is ignored if the percentage of the total registers in the design in the transitive fanout of the port exceeds a specified threshold percentage contained in the *ilm_ignore_percentage* variable. The default is 25.

You can use this option to help you identify the test enable and reset ports of your design. Before using it, examine the current value of the

ilm_ignore_percentage variable and reset it if necessary. Note that the *-auto_ignore* option might potentially ignore ports you do not want to ignore, or fail to ignore ports you want to ignore. Carefully read the messages issued by this command when you use this option to see which ports have been ignored and what percentage of registers to which they fanned out.

The *-ignore_ports* and *-auto_ignore* options are mutually exclusive.

`-latch_level` *`levels`*

Specifies the number of logic levels over which time borrowing can occur for latch chains that are a part of the interface logic. Substitute the number of levels you want for *levels*. By default, all latches found in interface logic are assumed to be potential borrowers. Thus, all logic from I/O ports to flip-flops or output ports are identified as belonging to interface logic. Note that this is the opposite of the default behavior of the *extract_model* command.

When you use this option, note that the value of *level* must account for the borrowing behavior of latches only on interface timing paths. If *n* represents a latch level, then *n* + 1 represents the number of latches maintained in latch chains that originate at input ports. The *n* + *1*th latch functions as an edge-triggered register and decouples I/O paths from internal paths. Use this option only when there are latches in the interface logic for a block. Specifying this option might potentially result in less accurate models, because not all latches are allowed to borrow. The *-context_borrow* and *-latch_level* options are mutually exclusive.

`-context_borrow`

Specifies that latch borrowing at the interface should be established based on the current context defined for the design. Therefore, latches that borrow based on arrival times defined on input ports and clocks defined on the design are traced through, but path tracing stops at nonborrowing latches. The *-context_borrow* and *-latch_level* options are mutually exclusive.

`-keep_ignored_fanout`

Specifies that the fanout from ignored input ports to interface logic is to be maintained in the model. Thus, ignored ports are not used to identify interface logic, but connections between ignored ports and interface logic are preserved. Using this option results in larger models, particularly in conjunction with the *-include_all_net_pins* option of the *write_ilm_netlist* command. Timing slacks for ignored ports might differ from those reported on the original netlist because connections from ignored ports to internal registers are not preserved in the model.

`-include_pins` *`pin_list`*

Specifies a list of pins that must be included in the interface logic. Substitute the list of pins you want for *pin_list*. This option can be used to optionally add internal pins to the interface logic. After the interface logic is determined, this list of pins is added to the interface logic. You can use this option to define additional interface logic pins that will otherwise be removed in the model. There will not be any affect on pins that are already in the interface logic. Use this option in conjunction with the *all_fanin* and *all_fanout* commands to include a particular cone of logic in the model.

`-critical_pins`

Specifies that critical pins interface logic must be identified. This option allows you to extract a model that keeps only the pins in the critical paths of the design.

You can use this option to extract a critical pins interface logic. The model generated contains the interface logic pins in the critical paths of the design. The model is compact compared to normal interface logic models, but might not be as accurate outside of the current context. This model can be used only to do critical path analysis. It might take much longer to generate this model than the normal model.

`-include_all_net_pins`

Specifies that all pins on nonclock interface logic nets and propagated clock interface logic nets are to be included in the netlist. Use this option if the interface logic model (ILM) is used in a non-SDF flows and delay calculations are performed using the information contained in the ILM. Preserving all pins on a net maintains correct pin capacitance information for the net.

The *-include_all_net_pins* option does not affect nonpropagated clock nets; that is, nets in the clock network that have user-specified source latency. This option is similar to the same option of the *write_ilm_netlist* command except that here it adds the extra pins to the interface logic. Therefore, if you query the interface logic pins by using the *get_ilm_objects* command, you notice these extra pins have been added.

The *write_ilm_netlist* command writes these pins only to the Verilog netlist, but does not add them to the interface logic. It is recommended that you use this option while extracting a critical pins interface logic model.

`-traverse_disabled_arcs`

Specifies that the interface logic should not be affected by disabled arcs/pins on the design. If specified, this option forces the traversal of disabled arcs while determining interface logic.

Feedback

Description

Sets the *is_interface_logic_pin* attribute on pins of the current design that are part of its interface logic. This is the first step in the generation of an ILM.

The interface logic on a block contains all cells whose timing is affected by, or affects, the external environment of a block. The following list describes such parts of the interface logic:

- All cells in timing paths that lead from input ports to registers or output ports that terminate the paths.
- All cells in timing paths that lead to output ports from registers or input ports that originate the paths.
- Clock trees that drive interface registers; including any registers in the clock tree. Clock-gating circuitry is part of interface logic if it is driven by external ports, but not if it is driven by registered outputs on a block.

Notice that interface logic does not include internal register-to-register paths and logic on a block associated only with these paths.

This command implicitly performs an update_timing on the design if required.

You can review the objects you have identified as interface logic by using the *get_ilm_objects* command. When you are satisfied with the interface logic designations, you can nondestructively generate a flattened Verilog netlist for the interface logic model (ILM) by using the *write_ilm_netlist* command. To generate script and back-annotation files for the ILM, use the *write_ilm_script*, *write_ilm_parasitics*, and *write_ilm_sdf* commands.

You can reset the *is_interface_logic_pin* attribute by executing the *identify_interface_logic* command again. You can remove all user-defined attributes, including the *is_interface_logic* attribute by using the *reset_design* command.

Examples

The following example places the *is_interface_logic_pin* attribute on all nonclock pins in the current design, except for pins in the fanin/fanout of ports *port1*, *port2*, and *port3*.

```
pt_shell> identify_interface_logic \\
-ignore [get_ports {port1 port2 port3}]
```

The following example additionally includes three levels of logic from internal pin *ff/Q*.

```
pt_shell> set ilm_pins [all_fanout -levels 3 \\
-flat [get_pin ff/Q]]
pt_shell> identify_interface_logic \\
-include_pins $ilm_pins
```

The following example extracts a critical pins interface logic model.

Feedback

```
pt_shell> identify_interface_logic -critical_pins \\
-include_all_net_pins
```

The following example places the *is_interface_logic_pin* attribute on all nonclock pins in the current design, except for pins identified by the *-auto_ignore* option. Because the value of the *ilm_ignore_percentage* variable is currently 35, pins are ignored if the percentage of the total design registers in their transitive fanout is greater than 35. The *-latch_level* option with a value of *1* states that the first latch encountered in I/O paths might potentially borrow, but the second latch can be treated as an edge-triggered device. Thus, two levels of latches are maintained in the interface logic.

```
pt_shell> printvar ilm_ignore_percentage
ilm_ignore_percentage = "35"
pt_shell> identify_interface_logic -auto_ignore \\
-latch_level 1
```

See Also

- get_ilm_objects
- write_ilm_netlist
- write_ilm_parasitics
- write_ilm_script
- write_ilm_sdf
- ilm_ignore_percentage

---

implement_eco

Performs ECO change implementation, including placement legalization, ECO routing, parasitics extraction, and static timing analysis.

Syntax

```
status implement_eco
  [-max_displacement_in_site_rows distance_in_minimum_site_rows]
  [-force]
```

Data Types

```
distance_in_minimum_site_rows       float
```

Arguments

`-max_displacement_in_site_rows distance_in_minimum_site_rows`

Specifies the maximum allowed displacement of cells that are moved, expressed as a multiple of the height of minimum-height site rows.

`-force`

Performs each implementation step, even if there are no ECO changes.

Description

The *implement_eco* command performs ECO change implementation, including placement legalization, ECO routing, parasitics extraction, and signoff-accurate static timing analysis, all in one command. This eliminates the need to pause PrimeTime after ECO operations and run different tools like IC Compiler II, StarRC, and IC Validator.

This command combines the ECO functions of multiple tools into a single command, allowing it to perform timing and physical co-optimization not possible when those tools are run separately, resulting in better QoR with less physical disturbance.

This command enables multiple iterations of incremental ECO without moving and transferring data between different tools, which is useful for manual ECO operations. When incremental extraction is enabled, the command records the current status and manages timing and physical data to enable incremental extraction and timing analysis. The incremental ECO is especially useful when the design is big but ECO changes are relatively few.

Requirements

These are the minimum requirements to run the *implement_eco* command in your sign-off environment:

- PrimeTime STA live or saved sessions
- NDM library path that matches your current PrimeTime session
- IC Compiler II and StarRC executable paths
- StarRC command file used for signoff extraction

To enable placement legalization, ECO routing, and extraction, you must specify the IC Compiler II and StarRC executables using the *set_implement_options* command. After you do this, run the *check_eco* command to initialize the tools and detect any environment setting errors. After the *check_eco* command runs successfully, you can run the *implement_eco* command.

Specifying Extraction Corners

For signoff-accurate parasitic extraction, you must specify the StarRC executable and StarRC command file used for extraction signoff. PrimeECO extracts the necessary information and uses it to perform incremental or full extraction while the *implement_eco* command is running.

If the PrimeTime session reads parasitic data files in SPEF format, you need to specify parasitic corners for each scenario. If the PrimeTime session reads parasitics files in GPD format, you need to set parasitic corners before you read the GPD parasitics.

The following script specifies the StarRC executable, command file, and parasitic corners. In this example, two scenarios named cold and hot have parasitic corners named cold.corner and hot.corner, respectively.

```
prompt> set_implement_options \\
  -starrc_exec StarXtract \\
  -starrc_cmd_files my_starrc.cmd \\
  -parasitic_corners {{cold cold.corner} {hot hot.corner}}
```

Specifying the StarRC Command File

PrimeECO uses a StarRC command file as a reference to create a new StarRC command file for extraction. It updates input and output data such as the design block name and parasitics file locations as specified in the *set_implement_options* command, but keeps other information such as the accuracy setting.

Lines ignored in original StarRC command file:

```
STAR_DIRECTORY
GPD
ECO_MODE
NETLIST_INCREMENTAL
NDM_DATABASE
BLOCK
COUPLING_REPORT_FILE
SUMMARY_FILE
```

New StarRC lines used for RC extraction:

```
NDM_DATABASE
BLOCK
ECO_MODE: YES
NETLIST_INCREMENTAL: YES
GPD
NETLIST_FORMAT: GPD
```

Other lines in the original StarRC command file are left unchanged.

In addition, PrimeECO supports the specification of multiple StarRC command files. This is needed when different command files are required across the set of the parasitic corners needed for static timing analysis. For example, if there are three parasitics corners named cold.corner, hot.corner and typical.corner, two command files can be put together as follows:

- "starrc.typical.corner.cmd" file contains the line

  ```
  SELECTED_CORNERS: typical.corner
  ```

- "starrc.other_corners.cmd" file contains the line

  ```
  SELECTED_CORNERS: cold.corner hot.corner
  ```

Both StarRC command files can be specified for PrimeECO as follows:

```
prompt> set_implement_options \\
  -starrc_exec StarXtract \\
  -starrc_cmd_files "starrc.typical.corner.cmd starrc.other_corners.cmd"
```

StarRC Extraction Mode

PrimeECO performs full extraction by default to generate the most accurate results. If incremental extraction mode is enabled, PrimeECO performs incremental extraction for ECO changes followed by incremental static timing analysis, resulting in faster turnaround. Incremental extraction is especially useful for small manual ECO changes.

Timer and Placement Legalizer Co-Optimization

By accessing timing and physical data simultaneously, PrimeECO places and legalizes ECO cells, and minimizes possible displacement, while avoiding unexpected timing changes. If PrimeECO cannot find good places for ECO cells, it moves neighbor cells with positive slack to create more room. When non-ECO cells are moved, the legalizer and timer work together to avoid new timing violations. For example, cells with larger positive timing slack can move farther than those with less slack.

You can also specify the maximum allowed displacement to further control unexpected timing changes. For example, to limit cell movement to no more than 2.0 minimum site rows heights:

```
prompt> implement_eco -max_displacement_in_site_rows 2.0
```

This maximum displacement is a soft limit, in which the tool tries to honor but does not guarantee.

Setting a Custom Environment for Implementation

You can use special settings for physical implementation such as legalization and routing by specifying custom IC Compiler II scripts using the *set_implement_options* command. The PrimeECO tool runs the specified scripts before or after executing the specified steps.

For example, the following command specifies custom configuration scripts for IC Compiler II setup, legalizing, and routing:

```
prompt> set_implement_options \\
  -icc2_post_link_script ./my_icc2_config.tcl \\
  -icc2_pre_leglize_script ./my_legalize_config.tcl \\
  -icc2_pre_route_script ./my_router_config.tcl
```

When you run the *implement_eco* command, the tool sources the respective configuration scripts at the times indicated in the option names: after linking, before legalizing, and before routing.

Custom Placement Legalization and Routing Scripts

Instead of allowing PrimeECO to control placement legalization and routing, you can direct the tool to use your own custom scripts to perform these steps, as shown in the following example.

```
prompt> set_implement_options \\
  -icc2_eco_legalize_script ./my_legalize.tcl \\
  -icc2_eco_route_script ./my_route.tcl
```

Specifying the Working Directory

The working directory is where the PrimeTime, IC Compiler II, and StarRC tools exchange data and files. You specify this directory with the *-work_dir* option of the *set_implement_options* command.

The working directory has the following subdirectories:

- *eco* -- Contains PrimeTime saved sessions for the incremental flow, change lists for IC Compiler II, and so on.
- *icc2* -- Contains temporary files for IC Compiler II execution.
- *starrc* -- Contains subdirectories used by StarRC extraction, including StarRC logs and summary files
- *lef* -- Contains LEF physical data files; created only if the *set_eco_options* command did not specify a different directory with the *-physical_lib_path* option

Fault Tolerance

To save runtime, the tool has fault tolerance features that record executed steps so that they do not need to be executed again by the *check_eco* or *implement_eco* command after you fix an error.

For example, suppose that *implement_eco* runs legalization and ECO routing successfully, but parasitic extraction fails due to an incorrect command file specification. After the issue is fixed, when you run the *implement_eco* again, PrimeECO skips legalization and ECO routing, as they have been executed already, and resumes StarRC extraction right away.

Feedback

Filler Cells

PrimeECO handles filler cells at various steps of the physical signoff ECO flow:

- During physically aware optimization commands (*fix_eco_timing*, *fix_eco_drc*, and *fix_eco_power*)
- During placement legalization performed by *implement_eco*
- During automatic filler cell insertion performed by *implement_eco*, when enabled by the *set_implement_options -insert_fillers* option

By default, filler cells are automatically detected as described in *set_eco_options* man page "Filler Cells" section.

Using the *-filler_cell_names* option of the *set_eco_options* command, you can specify additional filler cell types to be considered by PrimeECO during the filler cell steps mentioned above.

Inserting Metal Fill

You can instruct PrimeECO to invoke the IC Validator In-Design feature of IC Compiler II to perform initial and incremental metal fill insertion.

Initial metal fill insertion takes place after the NDM block is opened during the execution of the *check_eco* command. The following command example shows how you can specify your own IC Compiler II script for metal fill insertion in PrimeECO.

```
prompt> set_implement_options \\
  -icc2_post_link_script ./my_icv_init_script.tcl
```

For details, see "Inserting Metal Fill With IC Validator In-Design" in the IC Compiler II User Guide.

Incremental metal fill insertion takes place during the execution of the *implement_eco* command after the ECO routing step. This allows the metal fill to be considered by parasitic extraction.

The following command example shows how you can specify your own IC Compiler II script for incremental metal fill insertion in PrimeECO.

```
prompt> set_implement_options \\
  -icc2_pre_extract_script ./my_icv_eco_script.tcl
```

For details, see "Incremental Metal Fill Insertion" in the IC Compiler II User Guide.

Saving the ECO Design

To check legalization and routing status of the modified design, use the *check_legality* and *check_routes* commands. After you are comfortable with the results, save the design

with the *save_block* command. For details, see the man pages for the *check_legality*, *check_routes*, and *save_block* commands.

Writing Out the Implementation Changes

To write out the logical and physical changes performed in the current PrimeECO session, use the *write_implement_changes* command. Output formats include standard interfaces like Verilog and Design Exchange Format (DEF) to describe ECO changes in incremental form.

Incremental ECO and QoR

The PrimeECO tool supports incremental ECO flows with the following steps:

- Save PrimeTime sessions before ECO
- Perform ECO
- Incremental placement legalization
- ECO routing
- Incremental or full extraction
- Restore pre-ECO sessions
- Import physical and parasitics changes
- Run timing analysis for the final timing check

When StarRC incremental extraction is enabled, extraction and timing updates are also incremental, resulting in further runtime reduction.

For signoff accuracy, use StarRC full extraction. For faster runtime, especially for manual ECO operations with relatively few changes, use StarRC incremental extraction.

Note that slight timing differences between incremental and full extraction are expected due to the nature of incremental extraction and incremental timing updates.

Examples

The following example is a typical basic flow script.

```
# Configure PrimeECO
set_eco_options \\
  -physical_icc2_lib my_NDM \\
  -physical_icc2_blocks CPU_after_route
set_implement_options \\
  -icc2_exec icc2_shell \\
  -starrc_exec StarXtract

# Initialize
check_eco
```

```
# Run ECO
fix_eco_timing
implement_eco
```

Upon completion, the *implement_eco* command issues a report on the runtime spent in each step:

```
> Elapsed Time:
> Implementation Steps                        Seconds
> ------------------------------------------------
> Preprocessing                                    15
> Legalization                                     92
> ECO routing                                      47
> Post route processing                            55
> Extraction                                       60
> Timing update                                   581
> ------------------------------------------------
> Total implementation elapsed time               850

# Check final timing and physical DRC
report_global_timing
check_legality
check_routes

# Save the block
save_block -as CPU_after_eco
```

The following is a typical manual incremental ECO session.

```
# Configure PrimeECO with incremental extraction
set_eco_options \\
  -physical_icc2_lib my_NDM \\
  -physical_icc2_blocks CPU_after_route
set_implement_options \\
  -icc2_exec icc2_shell \\
  -starrc_exec StarXtract \\
  -starrc_mode incremental

# Initialize
check_eco

# 1st manual ECO
report_global_timing
size_cell ...
insert_buffer ...
implement_eco

# 2nd manual ECO
report_global_timing
size_cell ...
insert_buffer ...
implement_eco
```

```
# 3rd manual ECO
report_global_timing
size_cell ...
insert_buffer ...
implement_eco

# Check final timing and physical DRC
report_global_timing
check_legality
check_routes

# Save the block
save_block -as CPU_after_manual_eco
```

The following script runs a custom legalization script and a custom ECO routing script.

```
# Configure PrimeECO
set_eco_options \\
    -physical_icc2_lib my_NDM \\
    -physical_icc2_blocks CPU_after_route
set_implement_options \\
    -icc2_exec icc2_shell \\
    -starrc_exec StarXtract
set_implement_options \\
    -icc2_eco_legalize_script ./my_legalize.tcl \\
    -icc2_eco_route_script ./my_route.tcl

# Initialize
check_eco

# Run ECO
fix_eco_timing
implement_eco

# Check final timing and physical DRC
report_global_timing
check_legality
check_routes

# Save the block
save_block -as CPU_after_custom_eco
```

The following example specifies supplemental filler cell types on top of automatically detected ones. All filler cells are treated as open sites by physically aware optimization and placement legalization. In addition, *implement_eco* performs incremental filler cell insertion before ECO routing.

```
# Configure PrimeECO
set user_filler_cells "FILL4TLL"
set_eco_options \\
  -physical_icc2_lib my_NDM \\
  -physical_icc2_blocks CPU_after_route \\
```

```
  -filler_cell_names $user_filler_cells
set_implement_options \\
  -icc2_exec icc2_shell \\
  -starrc_exec StarXtract \\
  -insert_fillers

# Initialize
check_eco

# Run ECO
fix_eco_timing
implement_eco
```

The following example shows how IC Validator InDesign is configured for initial track-based metal fill insertion during *check_eco*.

The my_icv_init_script.tcl script file consists of the following:

```
#
# Setup or run ICV
#   This script is called right after ICC2  open_block is executed
#
set env(ICV_HOME_DIR) /global/apps/icv_2019.06-SP1-2
set target_arch "LINUX.64"
set env(PATH) "$env(ICV_HOME_DIR)/bin/$target_arch:$env(PATH)"
signoff_create_metal_fill -track_fill generic
```

You can specify this file in PrimeECO to perform initial metal fill with IC Validator as follows:

```
#
# Run ICV before ECO
#
set_implement_options -icc2_post_link_script ./my_icv_init_script.tcl
check_eco
```

See Also

- check_eco
- check_routes
- report_implement_options
- reset_implement_options
- save_block
- set_eco_options
- set_implement_options
- write_implement_changes

Feedback

index_collection

Given a collection and an index into it, if the index is in range, create a new collection containing only the single object at the index in the base collection. The base collection remains unchanged.

Optionally, a second index can be passed which creates a new collection with the objects between the two indices in the base collection. Makes an implicit assumption that index<=index2

Syntax

collection *index_collection*

```
collection1
index
[index2]
[step]
```

Data Types

collection1	collection
index	index
index2	index
step	int

Arguments

collection1

Specifies the collection to be searched.

index

Specifies the index into the collection.

The index is either a simple integer to specify the offset from the start of the collection or the string "end" optionally followed by a negative integer to specify the offset from the end of the collection.

For the start offset case the value must range from 0 to *sizeof_collection* - 1. For the end offset case the negative integer (if specified) must range from 0 to -(*sizeof_collection* - 1).

index2

Specifies an optional second index into the collection.

Specifies a second index to create a collection of a range of objects from the base collection, defined by the two indices [index, index2] (boundaries inclusive)

Feedback

`step`

Specifies the step increment from index to index2. By default 1 is used.

Description

You can use the *index_collection* command to extract a single object from a collection. The result is a new collection containing only that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index.

The range of indices is from 0 to one less than the size of the collection. If the specified index is outside that range, an error message is generated.

Commands that create a collection of objects do not impose a specific order on the collection, but they do generate the objects in the same, predictable order each time. Applications that support the sorting of collections allow you to impose a specific order on a collection.

You can use the empty string for the *collection1* argument. However, by definition, any index into the empty collection is invalid. Therefore, using the *index_collection* command with the empty collection always generates the empty collection as a result and generates an error message.

Note that not all collections can be indexed.

Examples

The following examples from PrimeTime use the *index_collection* command to extract the first, second, last and penultimate object of a collection.

```
pt_shell> set c1 [get_cells {u1 u2 u3}]
{"u1", "u2", "u3"}
pt_shell> query_objects [index_collection $c1 0]
{"u1"}
pt_shell> query_objects [index_collection $c1 1]
{"u2"}
pt_shell> query_objects [index_collection $c1 0 1]
{"u1", "u2"}
pt_shell> query_objects [index_collection $c1 end]
{"u3"}
pt_shell> query_objects [index_collection $c1 end-1]
{"u2"}
pt_shell> query_objects [index_collection $c1 end-1 end]
{"u2", "u3"}
```

Feedback

See Also

- collections
- query_objects
- sizeof_collection

---

infer_switching_activity

Infers the switching activity on the drivers of the preset and clear pins of the current design.

Syntax

string *infer_switching_activity*

```
[-apply]
[-output file_name]
[-nosplit]
[-verbose]
```

Data Types

file_name string

Arguments

`-apply`

Applies the proposed switching activity annotation on the drivers of the preset and clear pins.

`-output file_name`

Writes the script to the specified file in ASCII format.

`-nosplit`

Prevents line splitting when the information exceeds the specified column width. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column's width, the next field begins on a new line starting with the correct column.

`-verbose`

Displays detailed information about the specified preset or clear pins. The current and the proposed switching activity annotation reported is applicable only to the drivers of the corresponding preset and clear pins.

Feedback

Description

Use the *infer_switching_activity* command to infer the switching activity on the drivers of the preset and clear pins. These pins can be asynchronous preset and clear, or synchronous preset and clear. The switching activity include simple toggle rate and static probability on the drivers of these pins.

Use the *-apply* option to apply the inferred switching activity values on the drivers of the pins. When you run the *update_power* command, the PrimePower tool uses the inferred values to calculate and report power consumption.

For statistics on the switching activity annotation on the current design, use the *report_switching_activity* command.

Examples

In the following example, the *infer_switching_activity* command is specified after reading in the design. This command displays the current and proposed switching activity annotation for the rst pin.

```
pt_shell> infer_switching_activity
Created by infer_switching_activity on Thu May  2 17:51:45 2013

------------------------------------------------------------------------
-------
                                  Current      Current    Proposed
 Proposed
                                  Static       Toggle     Static
 Toggle
Objects            Type          Probability  Rate       Probability
 Rate
------------------------------------------------------------------------
-------
rst                driver        None         None       1.00
 0.00
------------------------------------------------------------------------
-------
1
```

Use the *infer_switching_activity -verbose* command to display information about the receiver pin and the receiver pin types.

```
pt_shell> infer_switching_activity -verbose
Created by infer_switching_activity on Thu May  2 17:51:45 2013

------------------------------------------------------------------------
-------
        Current     Current Proposed    Proposed
        Static      Toggle  Static      Toggle               Receiver
Driver  Probability Rate    Probability Rate    Receiver     Type
------------------------------------------------------------------------
-------
```

Feedback

```
rst     None        None    1.00        0.00    cnt_reg[0]/RN
 async_clear_preset
                             1.00        0.00    cnt_reg[3]/RN
 async_clear_preset
                             1.00        0.00    cnt_reg[1]/RN
 async_clear_preset
                             1.00        0.00    cnt_reg[4]/RN
 async_clear_preset
                             1.00        0.00    cnt_reg[2]/RN
 async_clear_preset
-------------------------------------------------------------------------
-------
There are 5 receivers for rst.
1
```

See Also

- set_switching_activity
- reset_switching_activity
- report_power

---

insert_buffer

Inserts a buffer at one or more pins.

Syntax

string *insert_buffer*

```
[-libraries lib_spec]
[-inverter_pair]
[-new_net_names new_net_names]
[-new_cell_names new_cell_names]
[-all_mim_instances]
pin_or_port_list
lib_cell
```

Data Types

lib_spec	list
new_net_names	list
new_cell_names	list
pin_or_port_list	list
lib_cell	string

Feedback

Arguments

`-libraries` *`lib_spec`*

If this option is specified, PrimeTime resolves the *lib_cell* value from the libraries contained only in the *lib_spec*. Libraries are searched in the order in which they appear in the *lib_spec*. *lib_spec* can be a list of library names, or collections of libraries loaded into PrimeTime; the latter can be obtained using the *get_libs* command. You cannot specify this option if a full library cell name has been specified.

`-inverter_pair`

Indicates that a pair of inverting library cells is to be inserted instead of a single non-inverting library cell.

`-new_net_names` *`new_net_names`*

Specifies the net name to be given to the new net that PrimeTime inserts. This option can only be used if only one buffer or an inverter pair is being inserted. If one buffer is being inserted, you have to pass only one net name. If an inverter pair is being inserted, you have to pass two net names. These names can be any valid net names, but must be the leaf names, that is, not the hierarchical names. The new names must not contain embedded hierarchical separators. The new names must be unique in the current context (as specified by current_instance). If you use this option, you have to also use the *-new_cell_names* option.

`-new_cell_names` *`new_cell_names`*

Specifies the cell name to be given to the new cell that PrimeTime inserts. This option can only be used if only one buffer or an inverter pair is being inserted. If one buffer is being inserted, you have to pass only one cell name. If an inverter pair is being inserted, you have to pass two cell names. These names can be any valid cell names, but must be the leaf names, that is, not the hierarchical names. The new names must not contain embedded hierarchical separators. The new names must be unique in the current context (as specified by current_instance). If you use this option, you have to also use the *-new_net_names* option.

`-all_mim_instances`

Insertion of a buffer at one or more pins in one MIM instance is replicated in all instances of a MIM set.

`pin_or_port_list`

Specifies a list of pins or ports to buffer.

lib_cell

Specifies the name of the library cell to use as a buffer (or inverter if the *-inverter_pair* option is specified). The *lib_cell* option can be library cell object or the name of a library cell. The former can be obtained using the *get_lib_cells* command. The latter can either be the full library cell name, such as *lib_name/lcell_name* or the just the base name of the library cell, such as *lcell_name*. You cannot specify the *-libraries* option and explicitly specify the full library cell name. If you invoke PrimeTime with the *-multi_scenario* option, you must use only the library cell-base name. For more information, see the "RESOLVING LIBRARY CELLS" section.

Description

The *insert_buffer* command adds a buffer at one or more specified pins or ports. Like all other netlist editing commands, for the *insert_buffer* command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The result of the *insert_buffer* command is a collection of the newly inserted cells on success, or the empty collection (empty string) on failure.

A library cell is a buffer if it has a single input and any number of outputs, as long as for each output, the logic function is either input or !input.

Newly created cells are given a unique name beginning with "U" and ending with a number; newly created nets are given a unique name beginning with "net" and ending with a number.

If you do not want PrimeTime to generate automatic names, you can use *-new_net_names* and *new_cell_names* options to override.

The *insert_buffer* command with *-all_mim_instances"* option applies changes to all the instances of the MIM set.

Suppose a CPU module is instantiated four times in the TOP module as CPU1, CPU2, CPU3 and CPU4.

The following example shows *insert_buffer* command being used in one instance of a MIM set i.e CPU1 with *-all_mim_instances"* option. The changes are replicated to all the other MIM instances of the set i.e CPU2, CPU3, CPU4.

```
pt_shell> insert_buffer CPU1/do1/D BUF1D9 -all_mim_instances"
Inserted 'U1' at 'CPU1/do1/D' with 'unit_delay_cells/BUF1D9'
Inserted 'U1' at 'CPU2/do1/D' with 'unit_delay_cells/BUF1D9'
Inserted 'U1' at 'CPU3/do1/D' with 'unit_delay_cells/BUF1D9'
Inserted 'U1' at 'CPU4/do1/D' with 'unit_delay_cells/BUF1D9'
1
```

The *insert_buffer* command uses the following basic rules to check its arguments:

- Each pin/port must be in scope (at or below the current instance). For a description of some special scoping rules, see the "BUFFERING INSIDE BOUNDARY PINS" section.
- Each pin/port must be connected to a net.
- Bidirectional pins cannot be buffered.
- The *lib_cell* cannot be sequential.
- The *lib_cell* must be a buffer, as previously defined.
- Without the *-inverter_pair* option, the library cell must have a noninverting output. The first noninverting output is used.
- When the *-inverter_pair* option is specified, the library cell must have an inverting output. The first inverting output is used. In this case, two cells are inserted, preserving the logic of the path.
- This command is a rare case where the objects argument (in this case, *pin_or_port_list*) does not necessarily stand alone. Pins on the same net can be grouped.

For more information about grouping pins on the same net and for additional connection rules, see the "CONNECTING THE NEW BUFFER" section.

Although you can mimic buffer insertion using other commands, such as the *create_cell*, *create_net*, *disconnect_net*, and *connect_net* commands, it is much more efficient to use the *insert_buffer* command.

Connecting the New Buffer

The new buffer is connected according to the following rules:

- The output of the new cell is always the new net, and the input of the new cell is always the old net. For example, buffering e1/Z or e3/A as follows:

```
e1/Z --- old_net --- e3/A

creates the following:

e1/Z --- old_net --- U1 --- net1 --- e3/A
```

- If the pin is a load, it is disconnected from its net, and connected to the new net.
- If the pin is a driver, all loads on that net are disconnected from the old net and connected to the new net.

- When pins on the same net are specified, they are grouped, and one buffer is inserted. Loads and drivers are grouped separately.
- If the net is multidriven, all driving pins must be specified for the command to succeed. In the following example, you cannot specify only e1/Z; you must specify both e1/Z and e2/Z. However, for the loads in Circuit 1, any combination can be buffered.

```
e1/Z ---+                +--- e3/A
        |                |
        +----- net ------+--- e4/A
        |                |
e2/Z ---+                +--- e5/A

            Circuit 1
            ---------
```

- Certain parallel buffer cases are also examined more closely. In Circuit 2, you cannot subdivide the inputs of the parallel drivers. You must buffer the inputs of both cell1 and cell2.

```
+--- cell1 ---+
net1 ---+             +-- net2 ---
        +--- cell2 ---+

             Circuit 2
             ---------
```

BUFFERING INSIDE BOUNDARY PINS

When you specify insertion of a buffer at a pin on the boundary of a hierarchical block, the *insert_buffer* command inserts the buffer either inside or outside the hierarchical block, depending on the current hierarchical scope. To insert the buffer inside the hierarchical block, set the scope to that block using the *current_instance* command and then execute the *insert_buffer* command.

The buffer is inserted within the block to which the *current_instance* command is set. For an example of buffering inside boundary pins, see the EXAMPLES section.

Buffering and its Effects on Parasitics

When parasitics are present and buffer insertion is performed the parasitics will only be preserved under the following conditions. (Note: What applies for a single buffer also applies for an inverter pair)

Buffering load pins in the presence of parasitics.

When a single load pin connected to a net with parasitics is buffered, the parasitics are preserved on the net connected to the input of the inserted buffer. There are no parasitics on the net connected to the output of the inserted buffer.

Feedback

Buffer load pin e3/A as follows:

```
+-- e4/A
                    |
e1/Z --- old_net ---|-- e3/A
             ^      |
             |      +-- e5/A
             |
      parasitics here
```

```
results in the parasitics being kept on the old net
```

```
                    +-- e4/A
                    |
e1/Z --- old_net ---|-- U1 --- net1 --- e3/A
             ^      |           ^
             |      +-- e5/A    |
             |                  |
      parasitics here     no parasitics
```

Buffering more than a single load pin results in the parasitics on the net connected to the input of the inserted buffer being removed and a warning being issued. (See PARA-060)

```
Buffer load pins e3/A, e4/A and e5/A
```

```
                    +-- e4/A
                    |
e1/Z --- old_net ---|-- e3/A
             ^      |
             |      +-- e5/A
             |
      parasitics here
```

```
results in the parasitics being removed
```

```
                                    +-- e4/A
                                    |
e1/Z --- old_net --- U1 --- net1 ---|-- e3/A
             ^                ^     |
             |                |     +-- e5/A
             |                |
     no parasitics     no parasitics
```

Buffering driver pins in the presence of parasitics.

When a single driver pin is buffered, on a net with only one driver, the parasitics are moved to the net connected to the output of the inserted buffer. There are no parasitics on the net connected to the input of the inserted buffer.

```
Buffer driver pin e1/Z as follows
```

```
                    +-- e4/A
                    |
```

```
e1/Z --- old_net ---|-- e3/A
            ^       |
            |       +-- e5/A
            |
      parasitics here

results in the parasitics being moved to net1

                                    +-- e4/A
                                    |
e1/Z --- old_net --- U1 --- net1 ---|-- e3/A
            ^                ^      |
            |                |      +-- e5/A
            |                |
     no parasitics     parasitics move here
```

Buffering more than a single driver pin on a multiply driven net results in the parasitics being removed from the net connected to the output of the inserted buffer. (See PARA-060)

Buffer driver pin e1/Z, e2/Z and e6/Z as follows:

```
e2/Z --+               +-- e4/A
       |               |
e1/Z --|--- old_net ---|-- e3/A
       |       ^       |
e6/Z --+       |       +-- e5/A
               |
       parasitics here

results in the parasitics being removed

e2/Z --+                              +-- e4/A
       |                              |
e1/Z --|--- old_net --- U1--- net1 ---|-- e3/A
       |       ^                ^     |
e6/Z --+       |                |     +-- e5/A
               |                |
        no parasitics    no parasitics
```

Resolving Library Cells

If the *lib_cell* option has been specified in base name only format, such as without a library from which to resolve it from, PrimeTime resolves the library cell according to the following methodology:

- If the *-libraries* option is specified, PrimeTime searches for library cells in the libraries contained within the *lib_spec* only.
- Alternatively, if the *-libraries* option is not specified, PrimeTime searches for the library cell in the libraries contained within the *link_path* variable.

The first library cell that is found is used.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Feedback

Examples

Refer to Circuit 1 for the following example; e1/Z and e2/Z both drive *net*. The first command specifies a *lib_cell* value that is not a buffer; the command fails, and an error message is generated. The second command specifies a buffer for a *lib_cell*, but does not specify all driver pins on a multi-driven net. Again the command fails with an error message. The third command specifies a buffer for *lib_cell* and specifies all pins for the multi-driven net. This time the command succeeds, and creates the new cell U1 and the new net net1.

```
pt_shell> insert_buffer e1/Z class/AN2P
Error: Could not insert 'class/AN2P' -
    lib_cell is not a buffer. (NED-010)
Error: No changes made. (NED-040)

pt_shell> insert_buffer e1/Z class/B1I
Error: Could not insert a buffer on object 'e1/Z':
    Multi-driver net and not all driver pins specified. (NED-012)
Error: No changes made. (NED-040)

pt_shell> insert_buffer {e1/Z e2/Z} class/B1I
Information: Inserted 'U1' at 'e1/Z', 'e2/Z' with 'class/B1I'. (NED-046)
{"U1"}
```

Refer to Circuit 1 for the following example. In the first command, an inverting buffer is inserted on two of the loads of "net". Two new cells are created, U2 and U3, as well as two new nets, net2 and net3. The second and third commands show the use of the *report_cell* and *report_net* commands to display the connections. Excerpts from the reports are shown.

```
pt_shell> insert_buffer {e3/A e4/A} class/IV -inverter_pair
Information: Inserted 'U2' and 'U3' at 'e3/A', 'e4/A' with 'class/IV.
 (NED-046)
{"U2", "U3"}
pt_shell> report_cell -connections e4
****************************************
Report : cell
         -connections
****************************************

Connections for cell 'e4':
    Reference:          B1I
    Library:            class

    Input Pins          Net
    ----------------    ------------
    A                   net3

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

```
1

pt_shell> report_net -connections net3
****************************************
Report : net
         -connections
****************************************

Connections for net 'net3':

    Driver Pins              Type
    ------------             ----------------
    U3/Z                     Output Pin (IV)

    Load Pins                Type
    ------------             ----------------
    e4/A                     Input Pin (B1I)
    e3/A                     Input Pin (B1I)

1
```

The following example demonstrates buffer insertion inside the boundary of a hierarchical block. To buffer a boundary pin on the inside of the hierarchical block, you must set the current instance to that block, as shown. Otherwise, the *insert_buffer* command would insert the buffer at the top level, outside instance e1.

Note the method used to get the boundary pin from within the block.

```
pt_shell> current_instance e1
u1
pt_shell> insert_buffer [get_pins ./z1] class/B1I
Uniquifying 'e1' (UBLOCK) as 'UBLOCK_0'.
Information: Inserted 'U1' at 'e1/z1' with 'class/B1I'. (NED-046)
{"e1/U1"}
```

See Also

- current_instance
- get_pins
- remove_buffer
- report_cell
- report_net
- size_cell
- swap_cell

Feedback

- write_changes
- link_path
- PARA-060

---

is_false

Tests the value of a specified variable, and returns 1 if the value is 0 or the case-insensitive string *false*; returns 0 if the value is 1 or the case-insensitive string *true*.

Syntax

```
status is_false
value
```

Data Types

```
value       string
```

Arguments

value

Specifies the name of the variable whose value is to be tested.

Description

This command tests the value of a specified variable, and returns 1 if the value is either 0 or the case-insensitive string *false*. The command returns 0 if the value is either 1 or the case-insensitive string *true*. Any value other than 1, 0, *true*, or *false* generates a Tcl error message.

The *is_false* command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings *true* or *false*. When such variables are set to *true* or *false*, they cannot be tested in the negative in an *if* statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x FALSE
if { !$x } {
  set y TRUE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, *is_false* allows you to test both conditions safely.

Feedback

Examples

The following example shows the use of the *is_false* command:

```
prompt> set x TRUE
TRUE

prompt> if { ![is_false $x] } {
?         set y TRUE
?       }
TRUE

prompt>
```

See Also

- is_true

---

is_true

Tests the value of a specified variable, and returns 1 if the value is 1 or the case-insensitive string *true*; returns 0 if the value is 0 or the case-insensitive string *false*.

Syntax

```
status is_true
value
```

Data Types

```
value      string
```

Arguments

value

Specifies the name of the variable whose value is to be tested.

Description

This command tests the value of a specified variable, and returns 1 if the value is either 1 or the case-insensitive string *true*. The command returns 0 if the value is either 0 or the case-insensitive string *false*. Any value other than 1, 0, *true*, or *false* generates a Tcl error message.

The *is_true* command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings *true* or *false*. When such variables are set to *true* or *false*, they cannot be tested in the negative in an *if* statement by simple variable

substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x TRUE
if { !$x } {
  set y FALSE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, *is_true* allows you to test both conditions safely.

Examples

The following example shows the use of the *is_true* command:

```
prompt> set x FALSE
FALSE

prompt> if { ![is_true $x] } {
?         set y FALSE
?       }

FALSE

prompt>
```

See Also

- is_false

I

license_users

Lists the current users of the Synopsys licensed features.

Syntax

string *license_users*

```
[feature_list]
```

Data Types

```
feature_list     list
```

Feedback

Arguments

feature_list

Specifies a list of licensed features for which to obtain the information. If you do not use this option, all features are shown.

Description

This command displays information about all licenses, related users, and host names currently in use. This command works only when Network Licensing is enabled.

Examples

In this example, all users of the PrimeTime feature are displayed.

```
pt_shell> license_users PrimeTime

john@node2      PrimeTime
paul@node1      PrimeTime
george@node3    PrimeTime, PrimeTime
rstarr@node3    PrimeTime

4 users listed.
```

See Also

- get_license
- list_licenses
- remove_license

---

link

The *link* command, a synonym for the *link_design* command, exists in PrimeTime for compatibility with Design Compiler.

See Also

- link_design

---

link_design

Resolves references in a design.

Syntax

string *link_design*

```
[-verbose]
[-force]
[design_name]
```

Data Types

```
design_name     string
```

Arguments

`-verbose`

Displays verbose messages during link.

`-force`

Forces relinking of the design. If you omit this option, the tool does not relink a fully linked design.

design_name

Specifies the design to be linked; the default is the current design.

Description

The *link_design* locates all designs and library components that are referenced by the current design and links them to the current design. During linking, the tool loads all files specified by the *link_path* variable if they are not already in memory. Successful linking results in a fully instantiated design on which you can perform analysis.

By default, the case sensitivity of the link is determined by the source of the objects being linked. Although it is not recommended, you can change the default behavior by setting the *link_force_case* variable.

Automatic Loading of Designs and Libraries

If you set the *link_path* and *search_path* variables, you need to read only your top-level design and then link. The tool automatically finds and loads all other required designs and libraries.

In the following example, the newcpu.db top-level design uses the cmos.db library. Since the *link_path* variable specifies cmos.db, the *link_design* command loads cmos.db. As linking proceeds, the tools loads any required design that is not already in memory by searching the paths specified by the *search_path* variable. For example, the referenced BOX1 design is not in memory, so the tool searches for BOX1.db in the *search_path* variable and loads it.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmos"
/designs/newcpu/v1.6/dbs /libs/cmos

pt_shell> set link_path "* cmos.db"
* cmos.db
```

```
pt_shell> read_db newcpu.db
Loading db file '/designs/newcpu/v1.6/dbs/newcpu.db'
1

pt_shell> link_design newcpu
Loading db file '/libs/cmos/cmos.db'
Linking design newcpu...
Loading db file '/designs/newcpu/v1.6/dbs/BOX1.db'
Loading db file '/designs/newcpu/v1.6/dbs/BOX2.db'
Loading db file '/designs/newcpu/v1.6/dbs/padring.db'
Design 'newcpu' was successfully linked.
1
```

Automatic Linking

Many PrimeTime commands attempt to link the current design for you. For example, the *report_timing* command performs linking if the current design is not linked. Automatic linking occurs only if the design is completely unlinked, and not if the current design is partially linked and has unresolved references. If the current design is fully linked, there is no need for automatic linking, so it is not attempted.

You can disable automatic linking by setting the *auto_link_disable* variable to *true*. Because determining the need for linking takes little runtime, setting this variable is normally unnecessary. The variable provides compatibility with Design Compiler.

Unresolved References

If the *link_design* command fails to resolve one or more references, you need to correct the source of the problem and relink the design. Failures are typically caused by

- Missing libraries or designs
- Incorrectly specified *link_path* or *search_path* variable
- A file that is located in the path but not accessible by you

After making the necessary corrections, you can decide whether to do an incremental relink or an initial link, or whether to allow the too to create black boxes for the unresolved references.

If you can resolve the references by changing the *link_path* or *search_path* variable, you must relink the design.

The creation of black boxes is controlled by the *link_create_black_boxes* variable, which is set to *true* by default. Unless you set this variable to *false*, the tool automatically converts each unresolved reference to a black box (an empty cell with no timing arcs). The result is a completely linked design on which analysis can be performed (assuming there are no other unrecoverable link errors). To prevent unresolved references, set the *link_create_black_boxes* variable to *true*.

Feedback

Black box creation can fail when there are multiple conflicting references, usually with generic logic. For example, if SELECT_OP has two references, one with five pins and the other with 20 pins, the second black box might not be created, and the design might not link. PrimeTime does not support generic logic, except for GTECH; if a design contains such generic logic, remove the design or replace it with an empty design.

Sometimes, black box creation fails. This occurs when there are multiple conflicting references. This happens most often with generic logic. For example, if there are two references to SELECT_OP, one with five pins, and the other with 20 pins, it is likely that the second black box is not created and the design does not link. Other than GTECH, PrimeTime does not support generic logic. Designs containing such generic logic should be removed or replaced by empty designs.

Linking With Mismatches

By default, if there are pin mismatches between instance and reference, the *link_design* command issues errors and fails. If you set the *link_allow_design_mismatch* variable to *true*, the *link_design* command issues warnings and continues. This allows you to gather useful information even when part of a design is missing.

To report mismatches, use the *report_design_mismatch* command.

Performance Considerations

The tool starts linking with the top design along with any other loaded designs and libraries. During the link process, other necessary designs and libraries are loaded. When the link process completes successfully, it produces a design that can be analyzed.

All subdesigns used to build the linked design are removed from memory.

Examples

The following examples show how a design links with many of the different linker options. First, the link fails when a library cannot be found in the link path because of a typo in the search path.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmoss"
/designs/newcpu/v1.6/dbs /libs/cmoss
pt_shell> set link_path "* cmos.db"
* cmos.db
pt_shell> read_db newcpu.db
Loading db file '/design/newcpu/v1.6/dbs/newcpu.db'
1
pt_shell> link_design newcpu
Error: Can't read link_path file 'cmos.db' (LNK-001)
Linking design newcpu...
Loading db file '/designs/newcpu/v1.6/dbs/BOX1.db'
Loading db file '/designs/newcpu/v1.6/dbs/BOX2.db'
Loading db file '/designs/newcpu/v1.6/dbs/padring.db'
Warning: Unable to resolve reference to 'NR4' in 'newcpu'. (LNK-005)
```

```
Warning: Unable to resolve reference to 'ND2' in 'newcpu'. (LNK-005)
Warning: Unable to resolve reference to 'FD2' in 'newcpu'. (LNK-005)
Information: Design 'newcpu' was not successfully linked:
             16 unresolved references. (LNK-003)
```

Correcting the value of the *search_path* variable and doing an initial link completely links the design without black boxes.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmos"
/designs/newcpu/v1.6/dbs /libs/cmos
pt_shell> link_design newcpu
Loading db file '/libs/cmos/cmos.db'
Linking design newcpu...
Loading db file '/designs/newcpu/v1.6/dbs/BOX1.db'
Loading db file '/designs/newcpu/v1.6/dbs/BOX2.db'
Loading db file '/designs/newcpu/v1.6/dbs/padring.db'
Design 'newcpu' was successfully linked.
1
```

See Also

- current_design
- list_designs
- list_libs
- read_db
- read_ddc
- read_verilog
- report_design_mismatch
- auto_link_disable
- link_allow_design_mismatch
- link_create_black_boxes
- link_force_case
- link_path
- search_path

---

list_attributes

Lists currently defined attributes.

Feedback

Syntax

string *list_attributes*

```
[-application]
[-class class_name]
[-nosplit]
```

Data Types

```
class_name       string
```

Arguments

`-application`

Lists application attributes as well as user-defined attributes.

`-class class_name`

Limit the listing to attributes of a single class. Valid classes are design, port, cell, net, and so on.

`-nosplit`

Prevents line-splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

Description

The *list_attributes* command displays an alphabetically sorted list of currently defined attributes. PrimeTime attributes divide into two categories: application-defined and user-defined. By default, the *list_attributes* command lists all user-defined attributes.

Using the *-application* option adds all application attributes to the listing. Note that there are many application attributes. It is often useful to limit the listing to a specific object class using the *class_name*.

Examples

This is an example listing of some attributes defined with *define_user_attribute*.

```
pt_shell> list_attributes

****************************************
Report : List of Attribute Definitions
Design :
****************************************

Properties:
    A - Application-defined
    U - User-defined
```

```
    I - Importable from db (for user-defined)

Attribute Name   Object  Type      Properties  Constraints
-------------------------------------------------------------
attr_b           cell    boolean   U
attr_d           cell    double    U
attr_f           cell    float     U
attr_i           cell    int       U,I
attr_ir1         cell    int       U           0 to 100
attr_ir2         cell    int       U           >= 0
attr_ir3         cell    int       U           <= 100
attr_oo          cell    string    U           A, B, C, D
attr_s           cell    string    U
attr_s           net     string    U
```

This example limits the listing to net attributes only.

```
pt_shell> list_attributes -application -class net

****************************************
Report : List of Attribute Definitions
Design :
****************************************

Properties:
    A - Application-defined
    U - User-defined
    I - Importable from db (for user-defined)

Attribute Name         Object  Type      Properties  Constraints
-------------------------------------------------------------------
area                   net     float      A
attr_s                 net     string     U
ba_capacitance_max     net     float      A
ba_capacitance_min     net     float      A
ba_resistance_max      net     float      A
ba_resistance_min      net     float      A
base_name              net     string     A
full_name              net     string     A
net_resistance_max     net     float      A
net_resistance_min     net     float      A
object_class           net     string     A
pin_capacitance_max    net     float      A
pin_capacitance_min    net     float      A
total_capacitance_max  net     float      A
total_capacitance_min  net     float      A
wire_capacitance_max   net     float      A
wire_capacitance_min   net     float      A
```

Feedback

See Also

- define_user_attribute
- get_attribute
- remove_user_attribute
- report_attribute
- set_user_attribute

---

list_designs

Lists designs that have been read into PrimeTime.

Syntax

string *list_designs*

```
[-all]
[-only_used]
```

Arguments

`-all`

In addition to listing the designs in memory, the *-all* option lists the designs that are instantiated in the current design but are removed from memory.

`-only_used`

Lists only designs in use, including the current design, any design that is linked or partially linked, and any designs instantiated in a linked design.

Description

The *list_designs* command lists the designs that have been read into PrimeTime. When used without options, the commands lists only designs that are currently in memory.

The command output shows the status of each design with the following notation:

* - Design is the current design

L - Design is linked

N - Design is not in memory

l - Design is partially linked

Examples

The following example lists the designs in memory.

```
pt_shell> list_designs

Design Registry:
 *L AD4FULA         /abc/designs/top.db:AD4FULA
  N AD4PG           /abc/designs/add1.db:AD4PG
    ADD5A           /abc/designs/add2.db:ADD5A
    MULT1           /abc/designs/top.db:MULT1
```

The following example lists only the designs that are in use and in memory.

```
pt_shell> list_designs -only_used
Design Registry:
 *L AD4FULA         /abc/designs/top.db:AD4FULA
    ADD5A           /abc/designs/add2.db:ADD5A
    MULT1           /abc/designs/top.db:MULT1
```

If you remove the MULT1 design, you can display it using the *-all* option only.

```
pt_shell> remove_design MULT1
Removing design 'MULT1'...
1
pt_shell> list_designs -all -only_used
Design Registry:
 *L AD4FULA         /abc/designs/top.db:AD4FULA
    ADD5A           /abc/designs/add2.db:ADD5A
  N MULT1           /abc/designs/top.db:MULT1
```

See Also

- current_design
- link_design
- remove_design

---

list_key_bindings

Displays all the key bindings and edit mode of current shell session.

Syntax

status *list_key_bindings*

```
[-nosplit]
```

Arguments

`-nosplit`

Indicates that lines are not to be split when column fields overflow.

Feedback

Description

The *list_key_bindings* command displays current key bindings and the edit mode. To change the edit mode, set the *line_editing_mode* variable in either the .synopsys_pt.setup file or directly in the shell.

The text CTRL+K is read as `Control+K' and describes the character produced when the k key is pressed while the Control key is depressed.

The text META+K is read as `Meta+K' and describes the character produced when the Meta key is depressed, and the k key is pressed. The Meta key is labeled ALT on many keyboards. On keyboards with two keys labeled ALT (usually to either side of the space bar), the ALT on the left side is generally set to work as a Meta key. The ALT key on the right can also be configured to work as a Meta key or can be configured as some other modifier, such as a Compose key for typing accented characters.

If you do not have a Meta key, ALT key, or another key working as a Meta key, the identical keystroke can be generated by typing ESC first and then typing k. Either process is known as metafying the k key.

Alternative key bindings work only in vi alternate (command) mode.

See Also

- sh_enable_line_editing
- sh_line_editing_mode

---

list_libs

Lists libraries that are read into PrimeTime.

Syntax

string *list_libs*

```
[-only_used]
```

Arguments

`-only_used`

Lists only the libraries in use. A library is in use if a linked design links to library cells from the library.

Description

The *list_libs* command lists the libraries that are read into PrimeTime. You can filter unused libraries out of the display by using the *-only_used* option. A library is in use if a linked design links to library cells from the library.

The command output shows the status of each library with the following notation:

* - Main library for the current design. The main library is the first library in the link path which has a time unit.

M - Maximum library of a maximum/minimum library relationship created by the *set_min_library* command.

m - Minimum library of a maximum/minimum library relationship created by the *set_min_library* command.

Examples

The following example lists all libraries.

```
pt_shell> list_libs

Library Registry:
    bus1_lib      /u/lib/bus1_lib.db:bus1_lib
    tech1         /u/lib/tech1.db:tech1
```

After linking a design, the main library can be identified. Further, using the *-only_used* option limits the display to the libraries that were used to link the current design.

```
pt_shell> link_design top_flat
Linking design top_flat...

Designs used to link top_flat:
  <None>

Libraries used to link top_flat:
  tech1

Design 'top_flat' was successfully linked.
1
pt_shell> list_libs -only_used

Library Registry:
 *  tech1         /u/lib/tech1.db:tech1
```

See Also

- current_design
- link_design
- list_designs
- set_min_library

Feedback

list_licenses

Shows the licenses that are currently checked out.

Syntax

status *list_licenses*

Arguments

None.

Description

The *list_licenses* command lists the name and number of licenses that you currently have checked out.

In distributed multi-scenario analysis (DMSA), the command issued at the manager process returns the total count of licenses checked out for the analysis. If you run the command at a worker process, the command shows the licenses in use by the current scenario running at the worker process.

If scenarios are running sequentially because of a limited number of worker processes or licenses, running *list_licenses* at a worker process shows the instantaneous license count at the moment of command execution.

Examples

This example shows the output from the *list_licenses* command.

```
pt_shell> list_licenses

Licenses in use:
        PrimeTime (1)
        PrimeTime-SI (1)
```

See Also

- get_license
- license_users
- remove_license

lminus

Removes one or more named elements from a list and returns a new list.

Feedback

Syntax

list *lminus*

```
[-exact]
original_list
elements
```

Data Types

```
original_list     list
elements          list
```

Arguments

`[-exact]`

Specifies that the exact pattern is to be matched. By default, *lminus* uses the default match mode of *lsearch*, the *-glob* mode.

original_list

Specifies the list to copy and modify.

elements

Specifies a list of elements to remove from *original_list*.

Description

The *lminus* command removes elements from a list by using the element itself, rather than the index of the element in the list (as in *lreplace*). The *lminus* command uses the *lsearch* and *lreplace* commands to find the elements and replace them with nothing.

If none of the elements are found, a copy of *original_list* is returned.

The *lminus* command is often used in the translation of Design Compiler scripts that use the subtraction operator (-) to remove elements from a list.

Examples

The following example shows the use of the *lminus* command. Notice that no error message is issued if a specified element is not in the list.

```
prompt> set l1 {a b c}
a b c

prompt> set l2 [lminus $l1 {a b d}]
c

prompt> set l3 [lminus $l1 d]
a b c
```

The following example illustrates the use of *lminus* with the *-exact* option:

```
prompt> set l1 {a a[1] a* b[1] b c}
a a[1] a* b[1] b c

prompt> set l2 [lminus $l1 a*]
{b[1]} b c

prompt> set l3 [lminus -exact $l1 a*]
a {a[1]} {b[1]} b c

prompt> set l4 [lminus -exact $l1 {a[1] b[1]} ]
a a* b c
```

---

load_constraints

Reads in a script containing design constraints. In golden constraint reader, the mapped objects are used for defining constraint

Syntax

status *load_constraint*

```
[-scope instance_name]
file_name
```

Data Types

```
instance_name     string
file_name         string
```

Arguments

`-scope instance_name`

Specifies the scope where the constraints contained in the *file_name* file are to be executed. If you do not specify the *-scope* option, the tool uses the current scope.

file_name

Specifies the name of the file that contains the constraints to be read.

Description

The *load_constraints* command sets the scope to the specified instance and executes the set of UPF commands in the *file_name* file. Upon return, the current scope is restored to what it was before invocation. The commands are executed the same way as the *source* Tcl command.

Feedback

Examples

The following example loads the top.sdc script.

```
prompt> load_constraint top.sdc
1
```

See Also

- enable_golden_constraints_reader
- sdc_name_map
- source

---

load_distributed_design

Load design netlist and associated data from a scenario into the DMSA manager shell.

Syntax

status *load_distributed_design*

Description

The *load_distributed_design* command will load the appropriate image (decided internally) from live DMSA scenarios at workers. If no images are available then the appropriate commands needed to generate a scenario will be issued.

The following data is loaded by the load_distributed_design - Netlist - Timing arc traversal data - Physical location data (if available & needed) - Selected application variables

The data loaded does not include the timing state of the design, constraints, parasitics and other data restored by restore_session outside of DMSA.

Internally, the command checks if netlist signatures of all scenarios in the DMSA session are identical. If signatures are not identical, the command errors out and does not load design data. The command initiates a save_session on one of the scenarios and then loads he saved session into the DMSA manager process.

See Also

- cache_distributed_attribute_data

---

load_of

Gets the capacitance of a library cell pin. It is a DC Emulation command provided for compatibility with Design Compiler.

Feedback

Syntax

float *load_of*

```
        lib_pin
```

Data Types

```
lib_pin       string
```

Arguments

lib_pin

Specifies the name of the library cell pin, or a collection that contains the library cell pin, for which to get the capacitance.

Description

The *load_of* command returns the capacitance of the given library cell pin. The command exists in PrimeTime for compatibility with Design Compiler. Complete information about the *load_of* command can be found in the Design Compiler documentation. The supported method for getting the capacitance of a library cell pin is by using the *get_attribute* command with the *pin_capacitance* attribute.

See Also

- get_attribute

---

load_upf

Reads in a script in the Unified Power Format (UPF) format.

Syntax

status *load_upf*

```
[-scope instance_name]
[-version upf_version]
[-supplemental supp_file_name]
[-echo]
file_name
```

Data Types

```
instance_name     string
upf_version       string
supp_file_name    string
file_name         string
```

Feedback

Arguments

`-scope` *`instance_name`*

Specifies the scope where the UPF commands contained in the *file_name* file are to be executed. If you do not specify the *-scope* option, the tool uses the current scope.

`-version` *`upf_version`*

Specifies the version of UPF to which the file conforms. Allowed version is 1.1 (the default).

`-supplemental` *`supp_file_name`*

Specifies the supplemental UPF file name to be read. This option is only supported in Golden UPF mode.

`-echo`

Displays the UPF commands as they are executed. By default, the UPF commands are not echoed on the screen as they are executed.

`file_name`

Specifies the name of the file that contains the UPF script to be read. If you are not using the golden UPF mode, this argument is required. In golden UPF mode, this argument is optional and specifies the name of the golden UPF file.

Description

The *load_upf* command sets the scope to the specified instance and executes the set of UPF commands in the *file_name* file. Upon return, the current scope is restored to what it was before invocation. The commands are executed the same way as the *source* Tcl command.

Examples

The following example loads the top.upf script.

```
prompt> load_upf top.upf
1
```

See Also

- create_power_domain
- source

---

log_trace

Control creation of a replay log of traced commands.

Feedback

Syntax

string *log_trace* [-start file_name|-stop|-pause|-resume] [-log log_string] [-quiet]

string *file_name* string *log_string*

Arguments

`-start` *`file-Name`*

This option is mutually exclusive with -stop, -pause, and -resume. The argument is the pathname of the file to which the replay trace log is output. The option opens a file for output and begins logging command execution strings and application variable changes. Note that if the file already exists, the command overwrites the existing file. There can only be a single replay trace log output at any time.

`-stop`

This option is mutually exclusive with -start, -pause, and -resume. This option stops replay trace log output and closes the log file. Once stopped, the replay log cannot be repopened for appending.

`-pause`

This option is mutually exclusive with -start, -stop, and -resume. This option pauses replay trace log output. Execution of all commands and application variable settings are ignored and omitted from the log until output is resumed with -resume.

Note that pausing replay log output may render the resulting log unable to replay or incapable of replicating tool results.

`-resume`

This option is mutually exclusive with -start, -stop, and -pause. This option resumes replay trace log output.

`-log` *`log_string`*

This option outputs the given log_string, verbatim, to the replay trace log. The option may be used to explicitly log a command invocation or a comment. If the string is a comment, it must start with the Tcl comment character '#'.

`-quiet`

If given, this option causes the command to ignore error conditions such as an attempt to pause the trace log output before it is started. The command will exit quietly when it encounters an error condition.

Feedback

Description

The *log_trace* command performs operations related to creating a flat replay log of traced command execution and application variable changes. All options are optional. Upon successful completion, the command returns the pathname of the trace log file, if any, or an error message if the command fails.

Trace log output contains a comment header with the tool name and version, and the date and time when trace log output was started.

The resulting replay log is not 100% guaranteed to be able to replay or to reproduce tool results. One reason is that the tool execution environment may be different. If any operations are performed during log creation that would further compromise the ability to replay the resulting log or to replicate tool results, a warning message is appended to the log. Such operations including pausing log output.

Invoking the *log_trace* with no options will report the current status of logging: on, paused, or off, followed by the currently open log file name, if any.

Examples

The following example shows reporting of the log status before starting trace logging.

```
prompt> log_trace
off
```

The following example starts trace log creation.

```
prompt> log_trace -start ./tracelog.tcl
/an/absolute/path/.tracelog.tcl
prompt> log_trace
on: /an/absolute/path/.tracelog.tcl
```

The following example pauses, then resumes log creation.

```
prompt> log_trace -pause
/an/absolute/path/.tracelog.tcl
prompt> log_trace
paused: /an/absolute/path/.tracelog.tcl
prompt> log_trace -resume
/an/absolute/path/.tracelog.tcl
prompt> log_trace
on: /an/absolute/path/.tracelog.tcl
```

See Also

- set_trace_option
- get_trace_option
- annotate_trace

Feedback

ls

Lists the contents of a directory.

Syntax

string *ls* [*filename ...*]

```
string filename
```

Arguments

filename

Provides the name of a directory or filename, or a pattern which matches files or directories.

Description

If no argument is specified, the contents of the current directory are listed. For each *filename* matching a directory, *ls* lists the contents of that directory. If *filename* matches a file name, the file name is listed.

Examples

```
shell> ls *.db *.pt
test1.pt          c1.db          c3.db          c5.db
test2.pt          c2.db          c4.db          c6.db
```

m

man

Displays reference manual pages.

Syntax

```
string man
topic
```

Data Types

topic string

Arguments

topic

Specifies the subject to display. Available topics include commands, variables, and error messages

Feedback

Description

The *man* command displays the online manual page for a command, variable, or error message. You can write man pages for your own Tcl procedures and access them with the *man* command by setting the *sh_user_man_path* variable to an appropriate value. See the man page for the *sh_user_man_path* variable for details.

Examples

The following command displays the man page for the *echo* command:

```
prompt> man echo
```

The following command displays the man page for the error message CMD-025:

```
prompt> man CMD-025
```

See Also

- help
- sh_user_man_path

---

map_design_mode

Maps specified design modes to cell modes, paths, or both.

Syntax

status *map_design_mode*

```
[-from from_pin_list]
[-to   to_pin_list]
[-through through_pin_list]
design_mode
[cell_mode_list]
[instance_list]
```

Data Types

```
from_pin_list          list
to_pin_list            list
through_pin_list       list
design_mode            list
cell_mode_list         list
instance_list          list
```

Feedback

Arguments

`-from` *from_pin_list*

Specifies a timing path from a given pin of the design that is active only for the specified design mode. The given paths are inactive for other design modes which are in the same design mode group.

`-to` *to_pin_list*

Specifies a timing path to a given pin of the design that is active only for the specified design mode. The given paths are inactive for other design modes which are in the same design mode group.

`-through` *through_pin_list*

Specifies a timing path through a given pin of the design that is active only for the specified design mode. The given paths are inactive for other design modes which are in the same design mode group.

design_mode

Specifies a design mode that is mapped to cell modes or paths. The design modes on the list must have been previously created using the *define_design_mode_group* command.

instance_list

Specifies a list of cells in which the specified modes are mapped to the design mode. This list must be accompanied by the *cell_mode_list* option.

cell_mode_list

Specifies a list of cell modes which are mapped to the design mode.

Description

The *map_design_mode* command maps design modes to cells or paths. The design modes must have been previously created by the *define_design_mode_group* command.

To map a design mode to a cell mode, use this command:

```
map_design_mode cell_mode_list instance_list design_mode
```

If the design mode changes from inactive to active, then all cell modes mapped to the design mode also become active.

To map a design mode to a path, use this command:

```
map_design_mode -from from_pin_list -to to_pin_list
  -through through_pin_list design_mode
```

Feedback

When a path is mapped to a design mode and the design mode is set inactive, the path becomes a false path. If the design mode is subsequently set active, then the path is reset and no longer considered false.

To see a report of the design modes specified for the current design, use the *report_mode -type design* command. The report also shows the cell modes and paths mapped to each design mode.

Examples

The following example defines two design modes, DM1 and DM2, maps the cell mode READ to design mode DM1 (for instance Uram1), then removes the design mode DM1. Removing DM1 also removes the mapping to the active cell mode READ.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode READ Uram1 DM1
pt_shell> remove_design_mode DM1
```

See Also

- define_design_mode_group
- map_design_mode
- report_mode
- reset_mode
- set_mode

---

mem

Shows the total memory used by PrimeTime.

Syntax

integer *mem*

Arguments

None.

Description

The *mem* command shows the maximum memory usage in kilobytes (KB) of this instance of PrimeTime.

The memory usage of a multiprocess application, such as PrimeTime, reflects aggregate memory used across all its processes. Conceptually, this is the total amount of memory

pages used for the entire application instance, without double counting shared pages (such as shared libraries and executables) across the application instance.

The memory usage of distributed processes is not included. For remote PrimeTime process memory usage, use the *report_host_usage* command.

Examples

The following example shows the memory usage information.

```
pt_shell> mem
8092
```

See Also

- cputime
- report_host_usage

---

merge_models

Merges multiple timing models (in LIB format) together to be one.

Syntax

string *merge_models*

```
[-lib_files lib_file_names]
[-mode_names mode_names]
[-group_names mode_group_names]
-output file_name
[-formats format_list]
[-tolerance merge_tolerance]
[-pin_cap_tolerance merge_tol_of_pin_cap]
[-single_mode]
[-keep_all_arcs]
[-merge_pg_pin {rail_name, [vol_opt]}]
```

Data Types

```
lib_file_names          list
mode_names              list
mode_group_names         list
file_name               string
format_list             list
merge_tolerance         float
merge_tol_of_pin_cap    float
rail_opt_list           list
```

Arguments

`-lib_files` *`lib_file_names`*

Specifies a list of Synopsys internal library (.lib) files to read, compile, and merge. The files must have been extracted using the *extract_model* command. Substitute the list you want for *lib_file_names*.

`-mode_names` *`mode_names`*

Specifies a list of mode names for the list of models in one mode group. All mode names specified by all mode_names option must match the list of library files specified. Substitute the list you want for *mode_names*. This option can be used multiple times. This option cannot be used with *merge_pg_pin* option.

`-group_names` *`mode_group_names`*

Specifies the names of the mode groups to which the models belong. If you do not specify this option, then default assign serial name *etm_modes_1*, *etm_modes_2*,... This option cannot be used with *merge_pg_pin* option.

`-output` *`file_name`*

Specifies the name of the output file that is used to save the merged model. Substitute the name you want for *file_name*.

`-formats` *`format_list`*

Specifies the formats the merge model is written in, either db, lib, or both. Substitute the formats that you want for *format_list*.

`-tolerance` *`merge_tolerance`*

Specifies the float tolerance used to compare delay or transition values of arcs in the timing models. It defines the tolerance for floating point number comparison when the timing arcs in the models are matched and compared against each other to determine they are equal. Float numbers are considered equal if the difference between them is within the tolerance.

The default is 0.04. The values from the first .lib file listed in the *-lib_file* option is used in the merged model. Substitute the tolerance you want for *merge_tolerance*; however, the value must be greater than or equal to zero.

This option cannot be used with the *keep_all_arcs* option.

`-pin_cap_tolerance` *`merge_tol_of_pin_cap`*

Specifies the float tolerance used to compare the capacitance attribute values for the matching pins across the models. Float numbers are considered equal if the absolute difference between them is within the specified tolerance.

The default is 0.002. The values from the first .lib file listed in the *-lib_file* option is used in the merged model if equal within tolerance.

`-single_mode`

Forces one timing arc to have only one defined mode or forces there to be only one single mode per timing arc in the merged model. This is necessary for downstream tools that cannot handle more than one mode on a timing arc. This normally results in larger, less compact timing models.

This option cannot be used with the *keep_all_arcs* option.

`-keep_all_arcs`

This makes the *merge_models* option ignore the *merge_tolerance* and suppresses all arc merging, and it implies *single_mode*. Use this option when performing two independent merges and the resulting merged models must have the same number of arcs. Resulting models are very large. This option cannot be used with either the *single_model* or *merge_tolerance* option.

`-merge_pg_pin` *`rail_opt_list`*

Specifies a list of voltage corners and their *high/low* options. If *high/low* not specified, merge with *default*: merge with all voltage corners from first lib file. All non-specified voltage must be the same. This option can be used multiple times. This option cannot be used with either the *mode_names* or *group_names* option. When this option is used, it is considered option *single_model* also applies.

Description

Merges multiple timing models together to be one. The timing models must be in Synopsys internal library (.lib) format. This command reads in all the specified files to generate a list of timing models. These timing models are processed, matched, compared, and merged to create a single timing model. This model has moded timing arcs such that, in any given mode, the static timing behavior of the merge model is equivalent to one of the models merged. The final merged model is saved on disk in the specified files and requested formats. PrimeTime and Design Compiler recognize the merged model with moded timing arcs for performing timing analysis.

The *keep_all_arcs* argument should only be used when a timing model has been extracted under several different conditions and the results are to be merged into two or more models in situations where the number of arcs must match. For example, say that a model TEST is extracted with two modes A and B, operating conditions min and max. There are four extractions, min A, min B, max A and max B. Now you want to merge all min values for modes A and B to get a minimum model, and all max values to get a maximum model. If you were going to use the merged models as the minimum and maximum libraries of the *set_min_library* command, their arcs must match. Therefore, you should merge the models using the *keep_all_arcs* option.

Feedback

Examples

The following command reads in the LIB models specified by the sram_write.lib and sram_read.lib files and generates a merged model containing an "SRAM_MODE" mode group with two "READ" and "WRITE" component modes. Timing arcs existing only in the sram_write.lib file are put into the merged model and marked with "WRITE" mode. Timing arcs existing only in the sram_read.lib file are marked with "READ" mode. Timing arcs existing in both models are compared to each other by their delay/transition values. If they are within 0.1 time units of each other, they are considered equal, and only one is kept in the merged model with no mode marked on the arc (valid for both modes); otherwise both arcs are kept in the merged model with each arc marked with a proper mode depending on which model it came from. Because the command requests all formats, the merged model is saved as sram_model_lib.db in Synopsys database format (.db), and sram_model.lib in Synopsys internal library format (.lib).

```
pt_shell> merge_models -lib_files \\
  {sram_write.lib sram_read.lib} \\
  -mode_names {WRITE READ} -mode_group SRAM_MODE \\
  -output sram_model -format {db lib} \\
  -tolerance 0.1
```

The following command reads in the LIB models specified by the sram_write.lib and sram_read.lib files and generates a merged model containing an "SRAM_MODE" mode group with two "READ" and "WRITE" component modes. Timing arcs existing only in the sram_write.lib file are put into the merged model and marked with "WRITE" mode. Timing arcs existing only in the sram_read.lib file are marked with "READ" mode. Timing arcs existing in both models are compared to each other by their delay/transition values. If they are within 0.1 time units of each other, they are considered equal, and only one is kept in the merged model with no mode marked on the arc (valid for both modes); otherwise both arcs are kept in the merged model with each arc marked with a proper mode depending on which model it came from. Because the command requests all formats, the merged model is saved as sram_model_lib.db in Synopsys database format (.db), and sram_model.lib in Synopsys internal library format (.lib).

```
pt_shell> merge_models -lib_files \\
  {etm_1.lib .lib} \\
  -mode_names {WRITE READ} -mode_group SRAM_MODE \\
  -output sram_model -format {db lib} \\
  -tolerance 0.1
```

The following command reads in the LIB models specified by the sram_write.lib and sram_read.lib files and generates a merged model containing the higher voltage of VDD1 ,the lower voltage of VDD2 from sram_write.lib and sram_read.lb and the VDD3 from sram_write.lib.

```
pt_shell> merge_models -lib_files \\
  {sram_write.lib sram_read.lib} \\
  -merge_pg_pin {VDD1 high} \\
```

Feedback

```
-merge_pg_pin {VDD2 low} \\
-merge_pg_pin VDD3 \\
-output sram_model -format {db lib} \\
-tolerance 0.1
```

See Also

- extract_model
- set_min_library

---

merge_saif

Reads a list of SAIF files with their corresponding weights, annotates switching activity attributes with merged toggle_rate, glitch_rate, and merged static_probability for nets, pins, ports, and power arcs in the current instance, and generates a merged output SAIF file.

Syntax

```
status merge_saif
   -input_list saif_file_and_weight_list
   -strip_path inst_name
   [-path prefix]
   [-ignore ignore_name]
   [-exclude exclude_file_name]
   [-derate_glitch value]
   [-quiet]
   [-output merged_saif_name]
   [-simple_merge]
```

Data Types

saif_file_and_weight_list	list
inst_name	string
prefix	string
ignore_name	string
exclude_file_name	string
value	float
merged_saif_name	string

Arguments

`-input_list` *saif_file_and_weight_list*

Specifies the name and the corresponding weight of the SAIF file list. *saif_file_and_weight_list* contains a list of { -input name.saif -weight number }, where "-input" and "-weight" are keywords. In addition, 0 < number < 100, and the sum of all weights is equal to 100. For example, { -input gate_back_1.saif -weight 20 -input gate_back_2.saif -weight 80 } means that

the files gate_back_1.saif and gate_back_2.saif are weighted by 20% and 80%, respectively.

`-strip_path` *`inst_name`*

Specifies the name as it appears in each SAIF file of the current design instance you want to annotate. The *merge_saif* command annotates all subinstances in the hierarchy of the specified instance, as well as annotating the instance itself.

`-path` *`prefix`*

Specifies a relative path from the current design to the hierarchical low-level design for which the SAIF file has been created. By default, absolute path names are used. Use this option if the SAIF file refers to an object in a hierarchy. Also if you want to read the switching information for a portion of the design, this option can be used. For example, if the SAIF was generated for the whole design, and you want to put switching information only on the nets/ports on the boundary and hierarchically under instance add14, then *-path* add14 option can be used.

`-ignore` *`ignore_name`*

Specifies the name of an instance in the SAIF file for which switching activity is to be ignored. The *merge_saif* command ignores switching activity within the hierarchy of that instance in the SAIF file.

`-exclude` *`exclude_file_name`*

Specifies the name of a file that contains a list of names to be ignored. *exclude_file_name* is used when you want to ignore switching activity for several instances. The file must contain each *ignore_name* on a separate line, without the *-exclude* option. As with the *-ignore* option, the ignore names are recognized after the *-strip_path* argument.

`-derate_glitch` *`value`*

Specifies a default derating factor value to be used for inertial glitches in the SAIF file. If the *-derate_glitch* option is not specified, a default derating factor of 0.5 is used.

`-quiet`

Specifies quiet mode and for instance suppresses warnings on design objects in the SAIF file that could not be annotated on the design.

`-output` *`merged_saif_name`*

Specifies the name of the output merged SAIF file.

`-simple_merge`

Indicates that all SAIF files are to annotate 100% of the nets, ports and pins of the current instance, and the *merge_saif* command is to perform a simple merge

of SAIF file data (without propagating the switching activity of non-annotated objects). Use the *report_switching_activity* command to verify that each SAIF file annotates 100% of the current instance.

Description

The *merge_saif* command reads a list of SAIF files with weight, computes the merged toggle_rate, glitch_rate, and static_probability, and annotates the switching activity attributes toggle_rate, glitch_rate, and static_probability for nets, ports, pins, and power arcs of the current instance. In addition, the *merge_saif* command optionally generates a merged output SAIF file. To identify the instance (and corresponding subinstances) you want to annotate, use the *-strip_path* argument. To specify a default derating factor other than 0.5, use the *-derate_glitch* option.

Examples

The following example reads and merges weighted SAIF files, annotates switching activity for the current design, and generates an output SAIF file.

```
pt_shell> set SAIF_DIR /home/user/ptpx/saif
pt_shell> set SAIF1 ${SAIF_DIR}/gate_back1.saif
pt_shell> set SAIF2 ${SAIF_DIR}/gate_back2.saif
pt_shell> set SAIF3 ${SAIF_DIR}/gate_back3.saif
pt_shell> set SAIF4 ${SAIF_DIR}/gate_back4.saif
pt_shell> merge_saif -input_list \\
             " -input $SAIF1 -weight 10 \\
               -input $SAIF2 -weight 20 \\
               -input $SAIF3 -weight 30 \\
               -input $SAIF4 -weight 40 " \\
               -strip_path E/UUT -output merged_saif.saif
```

See Also

- get_switching_activity
- report_power
- report_switching_activity
- reset_switching_activity
- set_rtl_to_gate_name
- set_switching_activity
- write_saif

---

move_objects

Moves and rotates the specified objects.

Feedback

Syntax

status *move_objects*

```
[-delta delta]
[-from point]
[-to point]
[-x x]
[-y y]
[-rotate_by angle]
[-group]
[-force]
[-simple]
[object_list]
```

Data Types

```
object_list  collection or selection
delta        list of two floats: x and y
point        list of two floats: x and y
x            float
y            float
angle        float
```

Arguments

object_list

Specifies a collection or selection set containing objects to move. If this option is not specified, the global selection set is used.

-delta *point*

Specifies the distance in the x- and y-directions to move the objects from the object's current location. This option is mutually exclusive with the *-to*, *-from*, *-x* and *-y* options.

-from *point*

Specifies the reference point on the object, or the collection of objects to be moved, for the *-to* option. By default, the command uses the lower-left corner of the bounding box of the given object, or collection of objects, as the reference point. This option must be used together with the *-to* option.

-to *point*

Specifies the new location of the reference point for the object or objects. This option is mutually exclusive with the *-delta*, *-x* and *-y* options.

-x *float*

Specifies the x-coordinate for the given object or collection of objects. All objects are moved so that their left edge is placed at the specified x-coordinate. This option is mutually exclusive with the *-delta*, *-from* and *-to* options.

Feedback

`-y` *float*

Specifies the y-coordinate for the given object or collection of objects. All objects are moved so that their bottom edge is placed at specified y-coordinate. This option is mutually exclusive with the *-delta*, *-from* and *-to* options.

`-rotate_by` *angle*

Specifies the rotation angle of the objects.

The valid values are:

```
0, 90, 180, 270,
CW90, CW180, CW270, CCW90, CCW180, CCW270
FLIPX, FLIPY
R0, R90, R180, R270, MX, MXR90, MY, MYR90
```

The rotation value definitions are:

```
CW90    is 90 degrees clockwise
CW180   and CCW180 are 180 degrees
CW270   is 270 degrees clockwise
CCW90   is 90 degrees counterclockwise
CCW270  is 270 degrees counterclockwise
FLIPX   is the reflection about the x-axis
FLIPY   is the reflection about the y-axis
```

The following values are synonymous:

```
0,   R0
90,  CCW90,  R90
180, CCW180, R180
270, CCW270, R270,
FLIPX, MY
FLIPY, MX
```

`-group`

Specifies if rotation should be made around common pivot point. If omitted, the rotation is performed around the object centers.

`-force`

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

`-simple`

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

Feedback

Description

This command moves one or more specified objects to a given location skipping all fixed or locked objects. The command keeps pin/terminal on edge of parent block and updates its layer if corresponding edit setting is set. See the *set_edit_setting* command for details.

Snapping is done automatically using the global snap settings.

Examples

The following example moves all selected objects by delta (-100 100).

```
prompt> move_objects -to {-100 100} -from {0 0}
```

The alternative syntax uses collection to specify objects.

```
prompt> move_objects [get_selection] -to {-100 100} -from {0 0}
```

See Also

- change_selection
- get_selection
- rotate_objects
- snap_objects

O

---

open_drc_error_data

Opens an error data file so that you can query or modify the error data.

Syntax

```
collection open_drc_error_data
[-file_name file_name]
[-readwrite]
[-readonly]
[-checkonly]
```

Data Types

```
file_name string
```

Feedback

Arguments

`-file_name` *`file_name`*

Specifies the file path of an external error data file to open.

`-readwrite`

Opens the specified external error data file in read/write mode.

When you enable this mode, the tool can modify the contents of the error data file. If you open a file in read/write mode, it stays in that mode even if you later open it in read-only mode.

This is the default open mode for external error data files, which are opened with the *-file_name* option.

The *-readwrite*, *-readonly*, and *-checkonly* options are mutually exclusive; you can specify only one.

`-readonly`

Opens the specified external error data file in read-only mode. However, if you previously opened the file in read/write mode, the file remains in read/write mode.

When you enable this mode, the tool cannot modify the contents of the error data file.

The *-readwrite*, *-readonly*, and *-checkonly* options are mutually exclusive; you can specify only one.

`-checkonly`

Checks if the specified error data file is already open and returns a collection that contains the error data object if the file is open. If the file is not open, it returns an empty string.

The *-readwrite*, *-readonly*, and *-checkonly* options are mutually exclusive; you can specify only one.

Description

The *open_drc_error_data* command opens an error data file so that you can query or modify the error data.

- External error data files are stored on disk.

  To open an external error data file, specify its file path with the *-file_name* option.

  When you open an external error data file, by default, the tool opens the file in read/write mode. To explicitly specify the open mode, use the *-readwrite* or *-readonly* option. When you use these options, the *-readwrite* option has the highest precedence. If you

Feedback

open the error data the *-readwrite* option and then open it again with the *-readonly* option, the open mode remains read/write.If you first open the error data with the *-readonly* option, and then open it again with the *-readwrite* option, the open mode changes to read/write.

The *open_drc_error_data* command increments the open count for the error data, except when you use the *-checkonly* option. The *open_drc_error_data* commands must be balanced by an equal number of *close_drc_error_data* commands to close the error data object.

The command returns a collection that contains the opened error data file. If no file is opened, the command returns an empty string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example opens the external error data file named "./my_design_dppinassgn.err" in read/write mode and returns a collection that contains the error data file.

```
prompt> open_drc_error_data -file_name ./my_design_dppinassgn.err
{my_design_dppinassgn.err}
```

See Also

- close_drc_error_data
- create_drc_error_data
- get_drc_error_data
- remove_drc_error_data
- save_drc_error_data

p

parallel_execute

Specifies a list of commands to be executed in parallel and their output files.

Syntax

status *parallel_execute*

Feedback

```
[-commands_only]
[-compress]
concurrent_cmds
```

Data Types

```
concurrent_cmds     list
```

Arguments

`-commands_only`

Prints output of commands after all commands complete execution.

`-compress`

Compresses when writing to file. If redirecting to a file, this option specifies that the output should be compressed as it is written. The output file is in a format recognizable by "gzip -d". You cannot specify this option with the *-commands_only* option.

concurrent_cmds

The list of commands, if the option *-commands_only* is specified. Otherwise, it is a list of interleaved strings that are commands and their output files. When there are sufficient computing resources available, these commands are to be executed in parallel with their results written into the specified output files separately.

Description

This command is a parallel directive that instructs PrimeTime to execute the list of commands in parallel. The output of commands is either printed out (if the option *-commands_only* is specified) or written out to the specified log files. It enables you to exploit parallelism across unrelated PrimeTime Tcl commands.

The following syntax shows how to use the *parallel_execute* command:

```
update_timing -full
parallel_execute {
  report_cmd1  rpt_file1
  report_cmd2  rpt_file2
  ...          ...
  report_cmdN  rpt_fileN
}
```

Any post-update reporting and analysis commands are applicable.

The *parallel_execute* command honors the resource constraint specified by the *set_host_options* command with the *-local_process* and *-max_core* options.

When there are sufficient resources, executing multiple PrimeTime post update reporting and analysis commands in parallel can reduce the overall runtime of the

script, comparing to when those reporting commands are executed sequentially. The commands recommended for the *parallel_execute* command include many common post-update commands, such as the *report_clock_timing*, *check_timing*, *report_bottleneck*, *report_exceptions*, *report_min_pulse_width*, *save_session*, path-based analysis, *create_ilm*, and *extract_model* commands or even user-defined Tcl procedures that are custom reports based on these native PrimeTime reporting commands.

To get the best results, it is important to understand that the *parallel_execute* command should contain post update reporting and analysis commands. Any commands that cause changes to the computed timing data and trigger incremental update result in severe errors and script and command execution immediately stops.

The *parallel_execute* command honors the resource constraint specified by the *set_host_options* command with the *-local_process* and *-max_core* options.

Note that the number of commands allowed inside the *parallel_execute* command has no practical limit, and PrimeTime automatically and dynamically schedules these commands if there are more commands then number of computing resources (CPUs or cores) available.

Also note that the *parallel_execute* command is itself a blocking command, which means PrimeTime does not execute the next command following the *parallel_execute* command after finishing all the commands specified in the list. To get closest to optimal speed-up, it is recommended that the longer runtime commands be specified first.

Examples

In the following example, four cores are allocated on the local host to perform parallel execution. Next, after the *update_timing* command, the *report_timing*, *report_constraints*, *save_session* PrimeTime commands and user Tcl procedure *report_qor* are executed in parallel.

```
pt_shell> set_host_options -local -max_cores 4
  ...

pt_shell> update_timing
pt_shell> parallel_execute {
            {report_timing -max 10000 -nworst 10} rpt_tim.log
            {report_constraints -all_violators} rpt_cstr.log
            {report_qor} rpt_qor.log
            {save_session  design_dir/my_session} /dev/null
          }
```

In the second example, parallel_execute is used to evaluate and print out the output of commands *check_timing* and *report_analysis_coverage*.

```
pt_shell> parallel_execute -commands_only {
            {check_timing}
            {report_analysis_coverage}
```

```
          }
Output of command 'check_timing':
Information: Checking 'no_input_delay'.
Information: Checking 'no_driving_cell'.
Information: Checking 'unconstrained_endpoints'.
Information: Checking 'unexpandable_clocks'.
Information: Checking 'latch_fanout'.
Information: Checking 'no_clock'.
Information: Checking 'partial_input_delay'.
Information: Checking 'generic'.
Information: Checking 'loops'.
Information: Checking 'generated_clocks'.
Information: Checking 'pulse_clock_non_pulse_clock_merge'.
Information: Checking 'pll_configuration'.
check_timing succeeded.

Output of command 'report_analysis_coverage':
****************************************
Report : analysis_coverage
Design : counter
 ...
****************************************

Type of Check         Total              Met         Violated
 Untested
-------------------------------------------------------------------------
-------
setup                     5         5 (100%)         0 (  0%)          0 (
  0%)
hold                      5         4 ( 80%)         1 ( 20%)          0 (
  0%)
out_setup                 5         5 (100%)         0 (  0%)          0 (
  0%)
out_hold                  5         5 (100%)         0 (  0%)          0 (
  0%)
-------------------------------------------------------------------------
-------
All Checks               20        19 ( 95%)         1 (  5%)          0 (
  0%)
```

See Also

- redirect
- set_host_options

---

parallel_foreach_in_collection

Iterates over the elements of a collection in parallel.

Feedback

Syntax

status *parallel_foreach_in_collection*

```
[-checks]
itr_var
collections
body
```

Data Types

itr_var	string
collections	list
body	string

Arguments

`-checks`

Enables the variable checks.

itr_var

Specifies the iterator variable. During each iteration, *itr_var* is set to a collection of exactly one object.

collections

Specifies a list of collections over which to iterate.

body

Specifies a script to execute one time per iteration at a worker process.

Description

Similarly to the *foreach_in_collection* command, the *parallel_foreach_in_collection* command iterates over elements of a collection, but with the added constraints and benefits of parallelism. It improves core utilization during the execution of custom Tcl scripts by executing multiple independent iterations simultaneously at worker processes. The required arguments for the *parallel_foreach_in_collection* command are the same as the arguments for the *foreach_in_collection* command: an iterator variable, the collections over which to iterate, and a script to apply at each iteration.

The *parallel_foreach_in_collection* command

- Automatically gathers any stdout/stderr output produced in parallel at workers during the execution of the *body* script. It prints the output, sorted by iteration order at the manager. For best performance, keep printing statements to a minimum as printing is done as a serial process.
- Requires a linked design; it attempts to perform an automatic link if a design is loaded but not linked.

- Supports the built-in *break*, *continue*, *return*, and *exit* commands, with the same syntax as other standard loop iteration constructs.
- Honors the resource constraints specified by using the *set_host_options* command with the *-local_process* and *-max_core* options.
- Falls back to running sequentially if used in the manager in distributed multi-scenario analysis (DMSA) mode.
- Falls back to running sequentially if nested within another parallel command such as *redirect -bg*, *parallel_execute*, or *parallel_foreach_in_collection* itself.
- Falls back to running sequentially if used while parasitics are being read in the background.

Constraints

The *body* script argument is subject to the following constraints, some of which can be lifted by using the sibling command *post_eval*:

- Modifications cannot be performed on any collections specified by the *collections* argument (for example, modifications through the use of the *append_to_collection* command).
- Modifications cannot be performed on the iterator variable *itr_var* during the loop.
- The *body* script can read and modify variables and call procedures defined in the local or global scope. However, it is forbidden to do so in a way that would lead to dependencies between iterations. In particular, an iteration of *body* is not allowed to write to a variable from local or global scope if another iteration reads the same variable.

  Note: The *body* script is run in parallel, and dependencies between iterations can potentially cause incorrect or unpredictable results due to race conditions.
- The *body* script is subject to the same restrictions imposed on the script argument of the *parallel_execute* command; that is, no change to the design state is allowed. This includes editing the design, changing constraints, performing updates, and setting user-defined attributes. If you attempt to execute such commands, the tool issues error messages.

  Additionally, do not expect modifications to local and global variables from within the *body* script to persist after the loop is completed because the *body* script is executed at worker processes (likewise parallel_execute).

Sibling Command: post_eval

You can issue the *post_eval* command from within *parallel_foreach_in_collection body* to perform persistent changes to variables and the design state. The *post_eval* command

has a single script argument that is submitted for execution at the manager process. This script is executed sequentially, and in guaranteed iteration order.

For more information, see the man page of the *post_eval* command.

Recommended Usage

The *parallel_foreach_in_collection* command speeds up custom complex Tcl scripts by applying parallelism to collection iteration, both pre- and post-update.

Although most commands in PrimeTime are capable of utilizing multiple cores (such as *get_pins* and *get_timing_paths*), many commonly-used built-in Tcl commands in custom Tcl scripts are inherently sequential. Therefore, their execution utilizes only one CPU core even if there are multiple cores allocated to PrimeTime. The *parallel_foreach_in_collection* command improves performance and scalability by utilizing the available CPU cores.

To convert your scripts to use the *parallel_foreach_in_collection* command, identify existing *foreach_in_collection* loops for conversion with the following guidelines:

- The loop should have a large overall runtime and iterate over many objects. If the collection is too small, then an ideal work balance cannot be achieved, resulting in under-utilization of available CPU cores.
- The loop should not contain any heavyweight commands that completely change the design such as timing, noise, or power updates, or link_design. These commands are generally too compute-intensive to be in a loop, and they are forbidden in *parallel_foreach_in_collection*.
- Each iteration of the loop should have a relatively compute-intensive filtering, reporting, analysis, or decision-making component, followed by one or more simple components, such as setting an attribute or modifying a variable to aggregate the results. In general, these components map to the *body* and *post_eval* arguments, respectively.
- Lastly, each iteration of the loop should have reasonable memory requirements, because parallel execution of multiple iterations can increase total memory consumption proportionally to this increment.

After you identify a loop for conversion to *parallel_foreach_in_collection*, review and restructure the loop so that you can add the relevant *post_eval* statements at the right places. Use the following guidelines for conversion:

- Global and local variables that are only read can be left unchanged, but variables that are modified need to be identified and classified according to their purpose. Temporary variables can be modified in the scope of the *body*, but any variable that is expected to persist after *parallel_foreach_in_collection* is completed needs to be moved into a *post_eval* statement.
- If there is any dependency between variables used in different iterations, it must be eliminated by rewriting the relevant part of the script. If a dependency

cannot be removed, then the loop is no longer a candidate for conversion to *parallel_foreach_in_collection*.

- Any changes to the design state, setting of constraints or user-defined attributes must be moved into a *post_eval* statement. During this process, it is important to ensure that there are no dependencies on design changes across loop iterations.
- The resulting *post_eval* statements should be reviewed to make sure that they are kept as small and fast as possible. *post_eval* statements are executed serially and ultimately determine the scalability and performance of the whole loop.

Variable Checks

When variable checks are enabled with the *-checks* option, PrimeTime monitors access to all Tcl variables and detects a broad variety of problems related to the misuse of variables or violations of the recommended usage of *parallel_foreach_in_collection*.

The variable checks are designed to aid in migrating from *foreach_in_collection*; you still need to carefully review the Tcl script being converted.

There are four main categories of errors that can be issued by the variable checks. For more information about these errors, see the man pages for MC-201, MC-202, MC-203, and MC-204.

Examples

The following example computes the endpoint with most violators and the number of violators for this endpoint.

```
set endpoints [add_to_collection [all_registers -data_pins -async_pins]
                                 [all_outputs] ]
set most_violators 0
set ep_with_most_violators ""
parallel_foreach_in_collection -checks endpoint $endpoints {
  set paths [get_timing_paths -to $endpoint -nw 1000
                              -slack_lesser_than 0]
  set num_violators [size_of_collection $paths]
  post_eval {
    if { $num_violators > $most_violators } {
      set most_violators $num_violators
      set ep_with_most_violators $endpoint
    }
  }
}
```

See Also

- foreach_in_collection
- parallel_execute

Feedback

- post_eval
- redirect
- set_host_options

---

parse_proc_arguments

Parses the arguments passed into a Tcl procedure.

Syntax

string *parse_proc_arguments* -args *arg_list result_array*

```
list arg_list
string result_array
```

Arguments

`-args arg_list`

Specified the list of arguments passed in to the Tcl procedure.

`result_array`

Specifies the name of the array into which the parsed arguments should be stored.

Description

The *parse_proc_arguments* command is used within a Tcl procedure to enable use of the -help option, and to support argument validation. It should typically be the first command called within a procedure. Procedures that use *parse_proc_arguments* will validate the semantics of the procedure arguments and generate the same syntax and semantic error messages as any application command (see the examples that follow).

When a procedure that uses *parse_proc_arguments* is invoked with the -help option, *parse_proc_arguments* will print help information (in the same style as using *help -verbose*) and will then cause the calling procedure to return. Similarly, if there was any type of error with the arguments (missing required arguments, invalid value, and so on), *parse_proc_arguments* will return a Tcl error and the calling procedure will terminate and return.

If you didn't specify -help, and the specified arguments were valid, the array variable *result_array* will contain each of the argument values, subscripted with the argument name. Note that the argument name here is NOT the names of the arguments in the procedure definition, but rather the names of the arguments as defined using the *define_proc_attributes* command.

The *parse_proc_arguments* command cannot be used outside of a procedure.

Feedback

Examples

The following procedure shows how *parse_proc_arguments* is typically used. The argHandler procedure parses the arguments it receives. If the parse is successful, argHandler prints the options or values actually received.

```
proc argHandler args {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo "  $argname = $results($argname)"
  }
}
define_proc_attributes argHandler -info "argument processor" \\
  -define_args \\
  {{-Oos "oos help" AnOos one_of_string {required value_help {values {a
 b}}}}
   {-Int "int help" AnInt int optional}
   {-Float "float help" AFloat float optional}
   {-Bool "bool help" "" boolean optional}
   {-String "string help" AString string optional}
   {-List "list help" AList list optional}}
  {-IDup int dup AIDup int {optional merge_duplicates}}}
```

Invoking argHandler with the -help option generates the following:

```
prompt> argHandler -help
Usage: argHandler    # argument processor
        -Oos AnOos              (oos help:
                                 Values: a, b)
        [-Int AnInt]            (int help)
        [-Float AFloat]         (float help)
        [-Bool]                 (bool help)
        [-String AString]       (string help)
        [-List AList]           (list help)
```

Invoking argHandler with an invalid option causes the following output (and a Tcl error):

```
prompt> argHandler -Int z
Error: value 'z' for option '-Int' not of type 'integer' (CMD-009)
Error: Required argument '-Oos' was not found (CMD-007)
```

Invoking argHandler with valid arguments generates the following:

```
prompt> argHandler -Int 6 -Oos a
  -Oos = a
  -Int = 6
```

See Also

- define_proc_attributes
- help

Feedback

pop_sms_scenario

Ends the scope of the closest/most recent previous *push_sms_scenario* command.

Syntax

string *pop_sms_scenario*

Description

The *pop_sms_scenario* command ends the scope of the closest/most recent previous *push_sms_scenario* command. It has no effect if no previous *push_sms_scenario* commands were executed. This command is only available with SMVA/SMC analysis.

Examples

The following example specifies 3 voltage reference names {high,low,typ} for each one of the supply groups SN1 and SN2. PrimeTime automatically considers all the relevant combinations of voltage levels, in this case up to nine prior to the use of the *push_sms_scenario* command. The example shows a sequence of a *push_sms_scenario* command followed by a *pop_sms_scenario* command and their expected effect on subsequent commands. For more examples, please look at the *push_sms_scenario* command manual.

```
pt_shell> set_voltage_levels SN1 -reference_names { high low typ }
pt_shell> set_voltage -o SN1 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN1 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN1 -reference_name typ 0.9 -min 1.0
pt_shell> set_voltage_levels SN2 -reference_names { high low typ }
pt_shell> set_voltage -o SN2 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN2 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN2 -reference_name typ 0.9 -min 1.0
```

The following command considers all SMS scenarios.

```
pt_shell> report_timing
```

The following command pushes the SN1:low SMS scenario collection.

```
pt_shell> push_sms_scenario -sms_scenarios [get_sms_scenario
 -supply_group SN1 -reference_name low]
```

The following command considers the SN1:low SMS scenario collection specified with the previous most recent *push_sms_scenario* command.

```
pt_shell> report_timing
```

The following command pops the SN1:low SMS scenario collection, ending its corresponding scope. There are no outstanding pushed SMS scenario collections after

Feedback

this command, as shown using the *get_current_sms_scenario* command. The subsequent command considers all SMS scenarios.

```
pt_shell> pop_sms_scenario
pt_shell> report_timing
pt_shell> get_current_sms_scenario
0
```

See Also

- push_sms_scenario
- get_current_sms_scenario

---

post_eval

Submits a script for sequential execution in the context of a parallel iterator.

Syntax

status *post_eval*

```
        script
```

Data Types

script string

Arguments

script

Specifies a script that to be executed sequentially at the manager process.

Description

The *post_eval* command can be used in the context of a parallel iterator such as the *parallel_foreach_in_collection* command to submit a script for execution at the manager process.

You can use multiple *post_eval* statements per loop iteration, and they can be put inside other constructs (such as *if* statements). *post_eval* statements are executed sequentially, and in guaranteed loop iteration order.

post_eval also supports built-in *break*, *continue*, *return*, and *exit* commands, with the same syntax as other standard loop iteration constructs.

Feedback

post_eval lifts from some of the restrictions normally imposed on the *body* of parallel iterators, and therefore can be used to perform actions such as changing constraints, setting user-defined attributes, and modifying persistent variables to aggregate results.

However, *post_eval* is still forbidden from executing any heavyweight commands that completely change the design such as timing, noise, or power updates, and link_design. These commands are generally considered to be too compute-intensive to be in a loop in the first place. If you use these commands, the tool issues error messages.

It is an error to issue a *post_eval* command outside of the context of a parallel iterator.

Variable Transfer and Identification

The *post_eval* argument is parsed at the beginning of the loop to identify references to variables from the worker process. Only variable references beginning with a '$' can be identified, and comments are ignored.

Because of this parsing requirement, *post_eval* statements can only be issued directly from within *parallel_foreach_in_collection body* (not from called Tcl procedures). Similarly, while the script argument to *post_eval* can call Tcl procedures, the parsing mechanism for variable identification does not follow called Tcl procedures.

The value of each of these variables at the worker at the point of the call to *post_eval* is sampled and transfered to the manager so that, when the *post_eval* script is executed at the manager, references to these variables can be successfully evaluated.

All basic Tcl types are supported for variable transfer, including string, boolean, integer, long, float, double, list, array, and dict. Collections of some types can also be transfered.

Supported collection object classes are pin, net, cell, port, lib, lib_cell, lib_pin, design, clock, and path_group.

For best performance, keep *post_eval* statements as small and fast as possible. It is important to avoid excessive variable transfer and only transfer small lists and arrays.

See Also

- collections
- parallel_foreach_in_collection

---

power_reset_feedthrough_endpoints

Reset feedthrough instance list and restore it back to previous cell list.

Syntax

```
int power_reset_feedthrough_endpoints
```

Feedback

Arguments

There is no input argument to this command.

Description

This command resets the cell list for power analysis which is set by "power_set_feedthrough_nets" command. This should only be run in time based mode.

report_power_analysis_options command can be used to check the instance list of FT nets reset by the command.

The command restores the cell list for power analysis to list of cells which was set before feedthrough net power analysis was done.

Examples

The following example shows how to reset instances on all FT nets and restores it with previously set list.

```
pwr_shell> power_reset_feedthrough_endpoints
power_reset_feedthrough_endpoints
1
report_power_analysis_options
****************************************
Report : Power Analysis Options
Design : TOP
Version: O-2018.06-SP5-2-DEV-20200122
Date   : Sat Jan 25 03:04:54 2020
****************************************

Power Analysis Mode: Time_based

Option Name                    Value
--------------------------------------------------------------------------------
-------
exclude lib cells              undefined
cells                          U_TOP_INV1 U_TOP_RST_1
 U_TOP_CLK1 snps_PD_ADD__iso1_snps_add_out_0___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_1___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_2___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_3___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_4___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_5___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_6___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_7___UPF_ISO
 snps_PD_ADD__iso1_snps_add_out_8___UPF_ISO U_TOP_INV2 U_TOP_RST_2
 U_TOP_CLK2 feed_through_top_reg_2_ feed_through_top_reg_1_
cycle_accurate_clock           undefined
cycle_accurate_cycle_count     0
include                        all_without_leaf
include_groups                 undefined
waveform_format                fsdb
```

Feedback

```
waveform_output
sdpd_tracking                  disabled
sdpd_tracking_cells            undefined

1
```

See Also

- report_power_analysis_options
- power_set_feedthrough_endpoints
- update_power

---

power_set_feedthrough_endpoints

Find the power for instances on the feedthrough nets in the design.

Syntax

```
int power_set_feedthrough_endpoints
[list of start/end points of FT nets]
```

Data Types

pin_list list

Arguments

pin list

To set the start/end point pins of FT nets.

Description

This command should only be run in time based mode. update_power should be used to calculate power and report_power shows the power values computed for instances on FT nets.

The command dumps out the list of pins which are not annotated (either in VCD/FSDB, SSA, SCA) in the fanin cone and needed for power analysis. The file is generated during "update_power" and is called 'FT_net_activity.rpt'.

report_power_analysis_options command can be used to check the instance list of FT nets set by the command.

Examples

The following example shows how to set FT net power on a single FT net with one pair of start pin and end pin.

Feedback

```
pwr_shell> power_set_feedthrough_endpoints {I_ADD/add_out[8]
 I_MULT/feed_through_top_in}
power_set_feedthrough_endpoints {I_ADD/add_out[8]
 I_MULT/feed_through_top_in}
1
update_power
Information: Clock clk is selected as the reference clock for cycle
 accurate peak power analysis of the current design. (PWR-280)
Information: Running time_based power analysis... (PWR-601)
Information : FT nets activity report 'FT_net_activity.rpt' generated
 successfully
```

The following example shows how to set FT net power on two FT nets with two pairs of start and end pins.

```
pwr_shell> power_set_feedthrough_endpoints {I_ADD/add_out_reg_8_1/D
 I_MULT/mult_in_feedthrough_reg_2_/Q I_ADD/add_out[8]
 I_MULT/feed_through_top_in}
power_set_feedthrough_endpoints {I_ADD/add_out_reg_8_1/D
 I_MULT/mult_in_feedthrough_reg_2_/Q I_ADD/add_out[8]
 I_MULT/feed_through_top_in}
1
```

See Also

- report_power_analysis_options
- power_reset_feedthrough_endpoints
- update_power

---

preview

Evaluates its arg list without actually executing it, but option values are "stored through" if option value-tracking is enabled.

Syntax

preview arg ...

Arguments

arg

The arg list will be evaluated without being executed.

Description

The preview command (like the built-in Tcl command eval) takes one or more arguments, which together comprise a Tcl script containing one or more commands. The

preview command concatenates all its arguments together (like eval), and passes the concatenated string to the CCI/Tcl interpreter recursively.

The preview command passes its concatenated command string to the CCI/Tcl command evaluator as usual, but the special feature of the preview command is that the command will not actually be executed by preview. Instead only the early stages of CCI argument/option checking will happen for the command string. Specifically, standard CCI checking of option arguments is performed, as is "store through" of new current option values for options with current-value-tracking enabled (if option checking succeeded). The preview command thus provides a way to update current values of a command's options (which have current-value-tracking enabled) without actually executing the command.

Conceptually, the CCI preview command is in the same "family" as the Tcl built-in command eval - the difference is that eval executes its concatenated command string, while preview does not, but preview does implement CCI option checking and "store through" for current values of options (which have current-value-tracking enabled).

The advantage of the preview command (compared with a command for turning on and off preview mode) is that a preview command is guaranteed to finish/exit (thus guaranteeing that the preview behavior in the CCI command evaluator will get turned off as soon as the preview command exits). If instead it was necessary to have matching commands for turning on and off "preview mode", bugs in Tcl script code might cause the "turn off preview mode" call to be skipped - then the interpreter would be stuck in "preview mode" indefinitely (and we do not want to take the risk of such a serious bug happening). By having preview as a command, we greatly reduce the risk that the interpreter could get "stuck in preview mode".

Note that the preview command itself does not implement any additional echoing to output of its concatenated command.

About preview and mutually exclusive options: preview performs all option checking for the previewed command as if it was to be executed (even though the command will not be executed). Thus the simultaneous presence of multiple mutually exclusive options will generate an error under preview (and "store through" of current option values will be aborted). This can be regarded as both a feature (it's consistent with normal command execution) and a limitation (it makes it impossible to set current values for mutually exclusive options of a command from a single run of preview). To workaround this limitation, we also provide the set_command_option_value command (which does not do checks such as the mutual exclusion check) - see the man page for set_command_option_value.

A possible power user application: power users could put preview commands in their home directory Tcl startup files for their favorite dialogs/commands to "seed" initial current values for these dialogs. This may lower the need for automatically saving replay scripts (of preview commands for each command/dialog) on exit from the application. Note: the set_command_option_value command may be easier to use than preview for this purpose.

Feedback

Examples

Consider a command named foo with two integer options, -bar1 and -bar2, that have current-value-tracking enabled. In the following example, the "current values" being tracked for the -bar1 and -bar2 options are updated to 10 and 20 respectively:

```
prompt> preview foo  -bar1 10  -bar2 20
prompt> get_command_option_values -current -command foo
-bar1 10  -bar2 20

Note that foo command is not actually executed in this example.
```

See Also

- get_command_option_values
- set_command_option_value

---

print_message_info

Prints information about diagnostic messages that have occurred or have been limited.

Syntax

string *print_message_info*

```
[-ids id_list]
[-summary]
```

Data Types

id_list	list

Arguments

`-ids` *id_list*

List of message identifiers to report. Each entry can be a specific message or a glob-style pattern that matches one or more messages. If this option is omitted and no other options are given, all of the messages that have occurred or have been limited are reported.

`-summary`

Generate a summary of error, warning, and informational messages that have occurred so far.

Description

The *print_message_info* command enables you to print summary information about error, warning, and informational messages that have occurred or have been limited with

the *set_message_info* command. For example, if the following message is generated, information about it is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to summarize all recorded information about generated diagnostic messages. Much of this can be done using the *get_message_info* command, but you need to know the specific message ID. By default, the *print_message_info* command summarizes all of the information. It provides a single line for each message that has occurred or has been limited, and one summary line that shows the total number of errors, warnings, and informational messages that have occurred so far. If an *id_list* is given, only messages matching those pat- terns are displayed. If the *-summary* option is given, a summary is displayed.

Using a pattern in the *id_list* is intended to show a specific message prefix, for example, "CMD*". Note that this does not show all messages with that prefix. Only those that have occurred or have been limited are displayed.

By default, the limit column shows the limit set by the *set_message_info* command. However, if the limit is not set and this type of message is included in the *sh_limited_messages* variable, this column shows the default limit of the *sh_message_limit* variable, and a symbol '*' also appears to show that this limit is from the *sh_message_limit* variable. Similarly, if the message is limited by *sh_global_per_message_limit*, this will be indicated by the symbol '-' following the limit column.

Examples

The following example uses the *print_message_info* command to display a few specific messages.

```
shell> print_message_info -ids [list "CMD*" APP-99]

  Id              Severity         Limit    Occurrences   Suppressed
  ------------------------------------------------------------------
  CMD-005            Error         10000 -           10            3
  APP-99       Information             1            0            0
```

At the end of the session, you might want to generate some information about a set of interesting messages, such as how many times each occurred (which includes suppressions), how many times each was suppressed, and whether a limit was set for any of them. The following example shows how to use the *print_message_info* command in this way. The symbol '*' in the limit column of PARA-004 means that this message is in the *sh_limited_messages* variable and this limit is the *sh_message_limit* variable.

```
shell> print_message_info

 Id              Severity         Limit    Occurrences   Suppressed
 ------------------------------------------------------------------
 CMD-005            Error         10000 -           12            0
```

```
PARA-004        Warning           100 *            1            0
APP-027     Information           100            150           50
APP-99      Information         10000 -            1            0

Diagnostics summary: 12 errors, 151 warnings, 1 informational
```

Note that the suppressed count is not necessarily the difference between the limit and the occurrences, since the limit can be dynamically changed with the *set_message_info* command, or the limit is from the *sh_message_limit* variable.

The sorting is by Severity (Error, Warning, Information), then by Occurrences, in descending order, then by Id.

See Also

- get_message_info
- set_message_info
- suppress_message
- sh_global_per_message_limit
- sh_message_limit
- sh_limited_messages

---

print_proc_new_vars

Checks for new variables created within a Tcl procedure.

Syntax

string *print_proc_new_vars*

Arguments

The *print_proc_new_vars* command has no arguments.

Description

The *print_proc_new_vars* command is used in a Tcl procedure to show new variables created up to that point in the procedure. If the *sh_new_variable_message_in_proc* and *sh_new_variable_message* variables are both set to *true*, this command is enabled. If either variable is *false*, the command does nothing. The result of the command is always an empty string.

When enabled, the command issues a CMD-041 message for each variable created in the scope of the procedure so far. Arguments to the procedure are excluded.

For procedures that are in a namespace (that is, not in the global scope), *print_proc_new_vars* also requires that you have defined some aspect of the procedure using the *define_proc_attributes* command.

Important: This feature has a significant negative impact on CPU performance. This should be used only when developing scripts or in interactive use. When you turn on the feature, the application issues a message (CMD-042) to warn you about the usage of this feature.

Examples

Assuming that the appropriate control variables are set to *true*, the following commands show new variables created within procedures:

```
prompt> proc new_proc {a} {
              set x $a
              set y $x
              unset x
              print_proc_new_vars
            }

prompt> new_proc 67
=New variable report for new_proc:
Information: Defining new variable 'y'. (CMD-041)
=END=

prompt>

prompt> proc ns::p2 {a} {
              set x $a
              print_proc_new_vars
            }

prompt> define_proc_attributes ns::p2

prompt> ns::p2 67
=New variable report for ns::p2:
Information: Defining new variable 'x'. (CMD-041)
=END=
```

See Also

- define_proc_attributes
- sh_new_variable_message
- sh_new_variable_message_in_proc

Feedback

print_suppressed_messages

Displays an alphabetical list of message IDs that are currently suppressed.

Syntax

string *print_suppressed_messages*

Arguments

The *print_suppressed_messages* command has no arguments.

Description

The *print_suppressed_messages* command displays all messages that you suppressed using the *suppress_message* command. The messages are listed in alphabetical order. You only can suppress informational and warning messages. The result of *print_suppressed_messages* is always the empty string.

Examples

The following example shows the output from the *print_suppressed_messages* command:

```
prompt> print_suppressed_messages
No messages are suppressed

prompt> suppress_message {XYZ-001 CMD-029 UI-1}

prompt> print_suppressed_messages
The following 3 messages are suppressed:
        CMD-029, UI-1, XYZ-001
```

See Also

- suppress_message
- unsuppress_message

printenv

Prints the value of environment variables.

Syntax

string *printenv*

```
[variable_name]
```

Data Types

```
variable_name      string
```

Feedback

Arguments

variable_name

Specifies the name of a single environment variable to print.

Description

The *printenv* command prints the values of the environment variables inherited from the parent process or set from within the application with the *setenv* command. When no arguments are specified, all environment variables are listed. When a single *variable_name* is specified, if that variable exists, its value is printed. There is no useful return value from *printenv*.

To retrieve the value of a single environment variable, use the *getenv* command.

Examples

The following examples show the output from the *printenv* command:

```
prompt> printenv SHELL
/bin/csh

prompt> printenv EDITOR
emacs
```

See Also

- getenv
- setenv

---

printvar

Prints the values of one or more variables.

Syntax

string *printvar*

```
[pattern]
[-user_defined | -application]
```

Data Types

```
pattern      string
```

Feedback

Arguments

pattern

Prints variable names that match *pattern*. The optional *pattern* argument can include the wildcard characters * (asterisk) and ? (question mark). If not specified, all variables are printed.

-user_defined

Shows only user-defined variables. This option is mutually exclusive with the *-application* option.

-application

Shows only application variables. This option is mutually exclusive with the *-user_defined* option.

Description

The *printvar* command prints the values of one or more variables. If the *pattern* argument is not specified, the command prints out the values of application and user-defined variables. The *-user_defined* option limits the variables to those that you have defined. The *-application* option limits the variables to those created by the application. The two options are mutually exclusive and cannot be used together.

Some variables are suppressed from the output of *printvar*. For example, the Tcl environment variable array *env* is not shown. To see the environment variables, use the *printenv* command. Also, the Tcl variable *errorInfo* is not shown. To see the error stack, use the *error_info* command.

Examples

The following command prints the values of all of the variables:

```
prompt> printvar
```

The following command prints the values of all application variables that match the pattern *sh*a**:

```
prompt> printvar -application sh*a*
sh_arch                  = "sparcOS5"
sh_command_abbrev_mode   = "Anywhere"
sh_enable_page_mode      = "false"
sh_new_variable_message  = "false"
sh_new_variable_message_in_proc = "false"
sh_source_uses_search_path = "false"
```

The following command prints the value of the *search_path* variable:

```
prompt> printvar search_path
search_path              = ".  /designs/newcpu/v1.6  /lib/cmos"
```

Feedback

The following command prints the values of the user-defined variables:

```
prompt> printvar -user_defined
a                    = "6"
designDir            = "/u/designs"
```

See Also

- error_info
- printenv
- setenv

---

proc_args

Displays the formal parameters of a procedure.

Syntax

string *proc_args*

proc_name

Data Types

proc_name string

Arguments

proc_name

Specifies the name of the procedure.

Description

The *proc_args* command is used to display the names of the formal parameters of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command *info* with the *args* argument.

Examples

This example shows the output of *proc_args* for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }

prompt> proc_args plus
a b

prompt> info args plus
```

Feedback

```
a b

prompt>
```

See Also

- proc_body

---

proc_body

Displays the body of a procedure.

Syntax

string *proc_body*

proc_name

Data Types

proc_name string

Arguments

proc_name

Specifies the name of the procedure.

Description

The *proc_body* command is used to display the body (contents) of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command *info* with the *body* argument.

Examples

This example shows the output of *proc_body* for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }

prompt> proc_body plus
 return [expr $a  + $ b]

prompt> info body plus
 return [expr $a  + $ b]

prompt>
```

Feedback

See Also

- proc_args

---

purge_distributed_attribute_data

Free the attribute data stored in the manager process.

Syntax

string *purge_distributed_attribute_data*

```
[-class class_name]
[-attributes attribute_list]
[-all]
```

Data Types

```
class_name          string
attribute_list      list
```

Arguments

`-class class_name`

Specifies the type of object: *port*, *cell*, *pin*, *net*. Use this option to specify the name of the class/object.

`-attributes attribute_list`

Free only the specified list of attributes for each object in the manager process.

`-all`

Free all the attribute data stored in the manager process.

Description

The *purge_distributed_attribute_data* will delete the unwanted attribute values kept in the manager process. This can be done selectively for few attributes or all the attributes at once using -all option.

Examples

The following example is shows the usage of purge_distributed_attribute_data command.

```
pt_shell> purge_distributed_attribute_data -class pin -attributes
 {min_rise_slack max_rise_slack}
```

Feedback

push_sms_scenario

Specifies SMS scenarios for selective application to subsequent SMVA analysis aware commands.

Syntax

string *push_sms_scenario*

```
[-sms_scenarios sms_scenarios_list]
```

Data Types

```
sms_scenarios_list       collection
```

Arguments

```
-sms_scenarios sms_scenarios_list
```

Specifies a collection of SMS scenarios to consider. The specified SMS scenarios are used by subsequent commands which support the -sms_scenarios option. This collection is created by *get_sms_scenarios*.

Description

The *push_sms_scenario* command specifies a collection of SMS scenarios for selective application to subsequent SMVA/SMC analysis aware commands. Specifically, those SMS scenarios apply to commands supporting the -sms_scenario option and attributes supporting SMS scenario specific queries. The specified scenarios are ignored by analysis commands such as update_timing, and by ECO commands. This command scope ends at a subsequent matching *pop_sms_scenario* command. Multiple *push_sms_scenario* can be nested. The closest, or most recent, *push_sms_scenario* command applies to its subsequent commands. The *get_current_sms_scenario* command can be used to query the pushed SMS scenarios applicable in the current context. The command has no effect on commands which specify their locally applicable SMS scenario using the -sms_scenarios option. It is recommended for SMVA/SMC flows to ensure consistent use of matching pairs of *push_sms_scenario* and *pop_sms_scenario* commands. This command is only available with SMVA/SMC analysis.

Examples

The following example specifies 3 voltage reference names {high,low,typ} for each one of the supply groups SN1 and SN2. PrimeTime automatically considers all the relevant combinations of voltage levels, in this case up to nine prior to the use of the *push_sms_scenario* command. The example shows a sequence of nested *push_sms_scenario* commands and their expected effect on subsequent commands.

```
pt_shell> set_voltage_levels SN1 -reference_names { high low typ }
pt_shell> set_voltage -o SN1 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN1 -reference_name high 1 -min 1.1
```

```
pt_shell> set_voltage -o SN1 -reference_name typ 0.9 -min 1.0
pt_shell> set_voltage_levels SN2 -reference_names { high low typ }
pt_shell> set_voltage -o SN2 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN2 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN2 -reference_name typ 0.9 -min 1.0
```

The following command considers all SMS scenarios.

```
pt_shell> report_timing
```

The following command pushes the SN1:low SMS scenario collection.

```
pt_shell> push_sms_scenario -sms_scenarios [get_sms_scenario
 -supply_group SN1 -reference_name low]
```

The following command considers the SN1:low SMS scenario collection specified with the previous most recent *push_sms_scenario* command.

```
pt_shell> report_timing
```

The following command considers the SN1:high SMS scenario collection specified locally using the -sms_scenarios command option.

```
pt_shell> report_timing -sms_scenarios [get_sms_scenario -supply_group
 SN1 -reference_name high]
```

The following command pushes the SN1:typ SMS scenario collection.

```
pt_shell> push_sms_scenario -sms_scenarios [get_sms_scenario
 -supply_group SN1 -reference_name typ]
```

The following command considers the SN1:typ SMS scenario collection specified with the previous most recent *push_sms_scenario* command and applies it to all commands executed by script1.tcl which support the -sms_scenarios command option.

```
pt_shell> source script1.tcl
```

The following command considers the SN1:low SMS scenario collection specified locally using the -sms_scenarios command option, and applies it to all commands executed by script2.tcl which support the -sms_scenarios command option.

```
pt_shell> source -sms_scenarios [get_sms_scenario -supply_group SN1
 -reference_name low] script2.tcl
```

The following command pops the SN1:typ SMS scenario collection specified with the previous most recent *push_sms_scenario* command, ending its corresponding scope.

```
pt_shell> pop_sms_scenario
```

Feedback

The following command considers the SN1:low SMS scenario collection, whose scope is still valid since it was not yet poped with a matching *pop_sms_scenario* command, as shown using the *get_current_sms_scenario* command.

```
pt_shell> report_timing
pt_shell> get_attribute [get_current_sms_scenario] description
{SN1:low}
```

The following command pops the SN1:low SMS scenario collection, ending its corresponding scope. There are no outstanding pushed SMS scenario collections after this command, as shown using the *get_current_sms_scenario* command. The subsequent command considers all SMS scenarios.

```
pt_shell> pop_sms_scenario
pt_shell> report_timing
pt_shell> get_current_sms_scenario
0
```

See Also

- pop_sms_scenario
- get_current_sms_scenario

---

py_eval

Evaluate python code. This is an optional feature, not all applications support Python.

Syntax

string *py_eval* -file *<path>*

code [-verbose]

```
string <path>
string code
```

Arguments

`-file <path>`

Evaluate python code in the specified file. This option cannot be specified with code.

code

Evaluate python statements embed within a Tcl script. This option cannot be specified with -file.

Feedback

`-verbose`

Echo commands and display results during evaluation. This option is only legal when combined with -file.

Description

This command evaluates Python code in the application. The Python language uses the "snps" module to control tool behavior. That module is implicitly imported into the Python global name space.

When this command is run from within Python (i.e. cmd.py_eval(...)) the specified code (or file) is evaluated in the current module namespace. When evaluated from Tcl the code is evaluated in global namesspace.

The Python snps module

Synopsys applications add a module called "snps" into the embedded Python interpreter. The snps module defines classes an objects that are used to interact with application builtin commands. Please see the following man pages for more details:

snps.cmd

Python object that provides access to application commands.

snps.collection

Python class that provides access to Synopsys collection values.

snps.value

Python class that provides access to application command return values.

snps.redirct

Python class used to redirect application output.

The Python interactive console

When the app_var sh_language is set to "python", the interactive language of the application is Python instead of Tcl. In this mode all input must be a valid Python expression or statement. When the input language is Python, the prompt will include "-py" to indicate this. The following example shows how to change the input language from Tcl, into Python, and then back again. Notice the changes to the prompt display:

```
fc_shell> set_app_var sh_language python
fc_shell-py> cmd.set_app_var('sh_language', 'tcl')
fc_shell>
```

The interactive Python console has a few extensions compared to the plain Python interpreter:

history()

This command prints the interactive command history and is an alias for the cmd.history() command. The tool keeps a separate history for interactive Python and Tcl. In Python mode you may evaluate commands from the history using the traditional "!" syntax.

source(fileName)

This command is an alias for the cmd.py_eval(file=fileName) command.

Examples

Evaluate the Python file in the global namespace.

```
prompt> py_eval -file pyFile.py
```

Find pyfile.py in the Python sys.path. Creates a new module (and namespace). If code in pyFile.py depends on features of the snps module, those must be imported manually into the module namespace.

```
prompt> py_eval {import pyFile.py}
```

The return value of the py_eval can be set by assigning to the cmd.result property. In the following example the Tcl variable "x" holds the value "10".

```
prompt> set x [py_eval {cmd.result = 10}]
```

See Also

- sh_language

q

query_cell_instances

Queries the instances of a mapped cell within the active scope. This is UPF query command similar to the *get_cells -of_object* PrimeTime command.

Syntax

list *query_cell_instances*

```
[-domain domain_name]
ref_name
```

Feedback

Data Types

```
domain_name         string
ref_name            string
```

Arguments

`-domain domain_name`

Limits the search to the instances in the power domain specified.

ref_name

Specifies a reference name (library cell or design) that is used to locate an instance.

Description

The *query_cell_instances* command queries the instances of a mapped cell within the active scope.

See Also

- get_cells

---

query_cell_mapped

Returns cells that are mapped to a specified instance.

Syntax

```
string query_cell_mapped
  instance_name
```

Data Types

```
instance_name      string
```

Arguments

instance_name

Specifies the instance to query.

Description

The *query_cell_mapped* command returns cells that are mapped to the specified instance. This UPF query command is similar to the *get_lib_cells -of_objects* command.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Feedback

See Also

- get_lib_cells

---

query_net_ports

Returns a list of ports that are logically connected to the specified net.

Syntax

collection *query_net_ports*

```
[-transitive false | true]
[-leaf]
net_name
```

Data Types

```
net_name     string
```

Arguments

`-transitive false | true`

When you set the *-transitive* option to *true*, the *query_net_ports* command applies to the descendants of the elements. The default is *false*.

`-leaf`

Returns only leaf cell ports.

net_name

Specifies the net to be queried.

Description

The *query_net_ports* commands returns a list of ports that are logically connected to the specified net. If no ports are connected, a null string is returned. This is UPF query command.

By default, this command searches both nonleaf and leaf cell ports. To search only leaf cell ports, use the *-leaf* option.

See Also

- get_ports

---

query_objects

Searches for and displays objects in the database.

Feedback

Syntax

string *query_objects*

```
[-verbose]
[-truncate elem_count]
[-class class_name]
object_spec
```

Data Types

class_name	string
elem_count	int
object_spec	list

Arguments

`-verbose`

Displays the class of each object found.

By default, the command lists only the name of each object. When you use this option, the command lists prefixes the name of each object with its class, as shown in the EXAMPLES section.

`-truncate elem_count`

Truncates the display to *elem_count* elements.

By default, the command displays up to 100 elements. To change the number of elements displayed by the command, use this option. To see all elements, set the *elem_count* value to 0.

`-class class_name`

Specifies the class used to search for objects when an element in the *object_spec* argument is not a collection. Valid classes are application-specific.

If you do not specify this option, the command displays only those elements in the *object_spec* argument that are collections and displays an error message for any other elements, as shown in the EXAMPLES section.

object_spec

Specifies the objects to find and display. Each element in the list is either a collection or a pattern.

- When you specify a collection, the command displays the contents of the collection.
- When you specify a pattern, the command searches the database for objects of the class specified in the *-class* option. If you do not specify the *-class* option, the command issues an error message for the pattern.

Feedback

Description

The *query_objects* command finds and displays objects in the application's runtime database. The command does not have a meaningful return value; it displays the objects found and returns an empty string.

To control the number of elements displayed, use the *-truncate* option. If the display is truncated, the command prints an ellipsis (...) as the last element. If the default truncation occurs, the command displays a message that shows the total number of elements that would have displayed, as shown in the EXAMPLES section.

Although the output from the *query_objects* command looks similar to the output from any command that creates a collection, the *query_objects* command always returns an empty string.

The *query_objects* command displays the output in either a Legacy format or in a Tcl compatible format. The default output format varies by tool but you can control the format yourself by setting the *query_objects_format* variable. For example, to force Tcl-compatible output use this command:

```
prompt> set_app_var query_objects_format Tcl
Tcl
```

The EXAMPLES section shows output in both the Legacy and Tcl formats.

Examples

These following examples show the basic usage of the *query_objects* command in both Legacy and Tcl format.

```
prompt> set_app_var query_objects_format Legacy
Legacy
prompt> query_objects [get_cells o*]
{"or1", "or2", "or3"}
prompt> query_objects -class cell U*
{"U1", "U2"}
prompt> query_objects -verbose -class cell \\
?   [list U* [get_nets n1]]
{"cell:U1", "cell:U2", "net:n1"}
prompt> set_app_var query_objects_format Tcl
Tcl
prompt> query_objects [get_cells o*]
{or1 or2 or3}
pt_shell> query_objects -class cell U*
{U1 U2}
pt_shell> query_objects -verbose -class cell \\
?   [list U* [get_nets n1]]
{{cell U1} {cell U2} {net n1}}
```

When you omit the *-class* option, only those elements of the *object_spec* argument that are collections generate output. The other elements generate error messages.

Feedback

```
prompt> query_objects [list U* [get_nets n1] n*]
Error: No such collection 'U*' (SEL-001)
Error: No such collection 'n*' (SEL-001)
{"n1"}
```

When the output is truncated, the command prints an the ellipsis at the end of the display. For the following example, assume the default truncation is 5 (it is actually 100).

```
prompt> query_objects [get_cells o*] -truncate 2
{"or1", "or2", ...}
prompt> query_objects [get_cells *]
{"or1", "or2", "or3", "U1", "U2", ...}
Output truncated (total objects 126)
```

See Also

- collections
- get_cells
- get_clocks
- get_designs
- get_generated_clocks
- get_lib_cells
- get_lib_pins
- get_libs
- get_nets
- get_path_groups
- get_pins
- get_ports
- get_qtm_ports
- get_timing_paths

---

query_port_net

Queries the net logically connected to a specified port and returns the net name.

Syntax

list *query_port_net*

Feedback

```
[-conn high | low]
port_name
```

Data Types

```
port_name     string
```

Arguments

`-conn high | low`

Queries the *high* (the default) or *low* connection. You can use this option only if the specified port is not on a leaf cell.

port_name

Specifies the port name to query.

Description

The *query_port_net* commands returns the net logically connected to a port. If no net is connected, the command returns a null string.

This is UPF query command.

See Also

- get_nets

---

quit!

quits the application without posting an application exit dialog

Syntax

quit!

Arguments

None.

Description

This command is an alternative to the *quit* command. When called with the GUI up, the *quit* command posts an application exit dialog. To avoid having to "deal with" the exit dialog, the user can instead call the *quit!* command which is guaranteed to exit the application without posting a GUI exit dialog.

Examples

```
icc_shell> quit!
```

Feedback

See Also

- quit

---

quit

Exits the shell.

Syntax

```
string quit
```

Arguments

The *quit* command has no arguments.

Description

The *quit* command exits from the application. It is basically a synonym for the *exit* command with no arguments.

Examples

The following example exits the current session:

```
prompt> quit
```

See Also

- exit

---

r

---

read_aocvm

Reads advanced and parametric on-chip variation (AOCV/POCV) derating factor tables.

Syntax

status *read_ocvm*

```
[-quiet]
ocvm_file
```

Data Types

```
ocvm_file     string
```

Arguments

`-quiet`

Suppresses the SEL-004, SEL-005, and AOCVM-006 errors and warnings. If this option is not set, errors and warnings are generated when objects specified in the AOCV tables are not found in the current design or libraries.

`ocvm_file`

Specifies the name of the AOCV/POCV file.

Description

The *read_ocvm* command reads AOCV/POCV derate factor tables from a disk file. The tables are annotated onto one or more design objects. Allowed design object types are hierarchical cells, library cells and designs. The AOCV/POCV data is used to reduce pessimism and improve the accuracy of results.

When named AOCV/POCV table sets are specified on objects in the design using the *set_ocvm_table_group* command, a cell (or net) derives its AOCV/POCV deratings from the table set associated with the hierarchical cell (or design) that (a) fully contains the cell (or net), and (b) is the lowest level (closest ancestor) hierarchical cell (or design) with an associated AOCV table set. If no such hierarchical cell (or design) containing the cell (or net) is found, the unnamed AOCV/POCV table set is used. Note that a net inherits the AOCV/POCV derate set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV/POCV table set), the lowest level hierarchical cell that fully contains the net is not the block. Therefore, the net derating comes from an AOCV/POCV table set that contains the block and fully encloses the net.

When applying AOCV tables on multiple object types that apply to an arc, the tool uses the following priority, in decreasing order of precedence, to determine the table that applies to the arc:

1. Library cell table
2. Hierarchical cell table
3. Design table

Note: Where a named AOCV table set applies to an arc, the precedence rules above apply to tables that belong to the specified set.

If the *timing_pocvm_precedence* variable is set to *lib_cell_in_file* the precedence of applying POCV tables is consistent with the AOCV precedence. By default, a design level POCV coefficient file takes precedence over LVF data available in library.

Feedback

Tables are processed in the order that they appear in the AOCV/POCV file. The last table of a specific type {object_type, rf_type, delay_type, derate_type} to be defined for an object in the file overwrites previous tables defined for the same object of the same type.

The *read_ocvm* command can read binary and compressed binary AOCV files created using the *write_binary_aocvm* command. No additional arguments are required to read binary or compressed binary AOCV files.

Examples

The following example shows an AOCV derate file, test.aocvm, with a single table annotated on all library cells.

```
pt_shell> sh cat test.aocvm

version:        1.0

object_type:    lib_cell
rf_type:        rise
delay_type:     cell
derate_type:    late
object_spec:    my_lib/*
voltage:        1.02
depth:          1 2 3
distance:       100 1000
table:          1.20 1.10 1.08
                1.22 1.15 1.11

pt_shell> read_ocvm test.aocvm
```

See Also

- get_timing_paths
- remove_ocvm
- report_ocvm
- report_timing
- reset_ocvm_table_group
- set_aocvm_coefficient
- set_ocvm_table_group
- write_binary_aocvm

Feedback

read_context

Reads binary context information previously written by the *write_context -format gbc* command.

Syntax

string *read_context*

```
[-name mim_group]
directory
```

Data Types

```
mim_group     string
directory     string
```

Arguments

`-name mim_group`

Loads the context of the specified multiply instantiated module (MIM). The default name is HyperScale_default. The *name* must be associated with the one set in the top-level analysis with the *set_hier_config name* command for intended instance sets.

directory

Specifies the directory that contains the binary context.

Description

This command reads in binary context information previously written by the *write_context -format gbc* command. Context loading occurs in the subsequent *update_timing* command, so you should use the *read_context* command before you perform a timing update.

If you run the *read_context* command after *update_timing*, the next command that requests a timing update will trigger a full timing update.

Examples

The following *write_context* command writes out the binary context for block A1 in the ./A1BC directory. The subsequent *read_context* command reads the block context information from that directory and then uses the context information in the next timing update.

```
pt_shell> characterize_context A1
 ...
pt_shell> update_timing
 ...
pt_shell> write_context -format gbc -output A1BC A1
```

```
 ...

pt_shell> read_context ./A1BC
 ...
pt_shell> update_timing
 ...
```

See Also

- characterize_context
- report_context
- write_context

---

read_context_leakage_data

Reads boundary leakage power information from leakage information side files and edge property side files and enables boundary leakage(CNOD) analysis during power analysis in PrimePower.

Syntax

status *read_context_leakage_data*

```
[-boundary_leakage_files file_fmt]
[-libcell_sdside_files file_fmt]
[-enable_pxe]
[-check_legality]
```

Data Types

file_fmt	list

Arguments

`-boundary_leakage_files` *file_fmt*

Specifies the list of boundary leakage information side files. The boundary leakage information side files must be provided to enable boundary leakage analysis during power analysis flow.

`-libcell_sdside_files` *file_fmt*

Specifies the list of edge proptery side files. The edge proptery leakage side files must be provided to enable boundary leakage analysis during power analysis flow.

`-enable_pxe`

Enables PXE checking in boundary leakage analysis flow. The option is optional and is applicable only for CNOD spec version v0.9 and later versions.

Feedback

`-check_legality`

Enables detection of empty spaces in cell rows and also checks if any cell is present in the design that doesn't fits vertically in cell row(s). If any of these is detected, the detailed report in dumped in file legality_check_detailed.log in the same folder from which pt/power shell is launched.

Description

PrimePower supports boundary leakage power analysis and reporting for advanced design nodes. The *read_context_leakage_data* command reads boundary power leakage information from leakage information side files and edge property side files and uses it for boundary leakage(CNOD) analysis during power analysis. This command enables boundary leakage power analysis and reporting.

Design physical information is needed for boundary leakage power analysis and physiical information files - LEF and DEF must be read in through PrimeTime ECO *set_eco_options* and *check_eco* commands prior to running boundary leakage analysis.

Examples

The following example reads two boundary leakage side files and two edge information side files

```
pt_shell> read_context_leakage_data -boundary_leakage_files { leak_sd1
 leak_sd2 } -libcell_sdside_files { edge_sd1 edge_sd2 }
```

See Also

- set_eco_options
- check_eco
- read_edge_annotation
- write_edge_annotation

---

read_ctpm

Read CTPM file into current design for PrimeShield SPICE2Design feature.

Syntax

string *read_ctpm*

ctpm_file

Feedback

Data Types

ctpm_file string

Arguments

ctpm_file

Specifies the CTPM file to read in.

Description

The command *read_ctpm* is used in PrimeShield SPICE2Design feature. This feature enables analysis for timing and power impact assessment of a given target. This target can be defined through a spice model representing next PDK, spot model, different spice corner, etc.

Through ML, Project Sicily aims to bridge the gap between the base and the target by capturing and gap through spice process parameters placed inside of a Compact Timing Power Model (CTPM) database. When CTPM is consumed in a base STA session, the QoR of this session is shifted such that to match QoR of the target.

After generating the CTPMs, you can use them in a real design to improve the QoR, by reducing the gap between the base and the target PDK the CTPM generation was based on. The *read_ctpm* command reads CTPM generated by *gen_ctpm* into design.

To perform a static timing analysis using a generated CTPM, perform the following steps: 1) Enable the PrimeShield tool for the CTPM generation using the following settings: *set ps_enable_analysis true set ps_enable_spice2design_analysis true* 2) Define the mapping between the base library already loaded in the design and its augmented library. See the following example: *define_sensitivity_lib_mapping [get_lib lib1] -side_lib lib1_side_file.db* 3) Read CTPM generated based on target PDK into the design. See the following example: *read_ctpm lib1.ctpm* 4) Peform complete STA: *update_timing -full*

Examples

The following script uses CTPM generated from target SPICE PDK to improve STA QoR comparing to target PDK:

```
set link_path orig.db
read_verilog ...
link_design
read_parastics ...
read_sdc ...

set ps_enable_analysis true
set ps_enable_spice2design_analysis true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]
read_ctpm lib1.ctpm
update_timing -full
```

Feedback

See Also

- gen_ctpm
- report_ctpm
- ps_enable_spice2design_analysis

---

read_db

Reads in one or more design or library files in Synopsys database (db) format.

Syntax

string *read_db*

```
[-netlist_only]
[-library]
file_names
```

Data Types

file_names	list

Arguments

`-netlist_only`

For designs only; ignored for libraries. Indicates that only the netlist is to be read; attributes are not to be read. This can greatly increase performance while reading designs.

`-library`

Indicates that the file is a library db file; using this option optimizes the reading of large library db files.

file_names

Specifies names of one or more files to be read. Typically, only one file is read.

Description

Reads design and library information from Synopsys database (db) files into PrimeTime.

To locate files that have relative pathnames, the command uses the *search_path* variable and searches for each file in each directory listed in the *search_path* variable. Files that have absolute pathnames are loaded as though there were no *search_path* variable. To determine the file that the *read_db* command loads, use the *which* command.

Each file can contain one or more design objects. After the files are loaded, to view the design objects, use the *list_designs* and *list_libs* commands. The db file itself is used only

during the read process, and is transformed into the internal format of PrimeTime during the *read_db* command . To remove designs and libraries, use the *remove_design* and *remove_lib* commands.

Synopsys design db files are used for many tools and can contain many complex attributes. However, only certain attributes apply to PrimeTime (for example, clocks, back-annotation). By default, these attributes are loaded from the db file. However, all information can also be brought into PrimeTime through scripts or other file formats (for example, SDF). In some cases (most notably SDF), you can save CPU time and memory usage by reading the information from the alternate source. In these cases, use the *-netlist_only* option to read the db file. The *-netlist_only* option speeds up the read process and drastically reduces memory usage in the case of SDF back-annotation. You can then read the SDF file using the *read_sdf* command.

If the *power_enable_analysis* variable is set to true, the *read_db* command also loads the power models in library db files.

Examples

In the following example, the file newcpu.db is read based on the *search_path* variable.

```
pt_shell> set search_path "/designs/newcpu/v1.6/dbs /libs/cmos"
/designs/newcpu/v1.6/dbs /libs/cmos

pt_shell> read_db newcpu.db
Loading db file '/designs/newcpu/v1.6/dbs/newcpu.db'
1
```

See Also

- list_designs
- list_libs
- read_sdf
- remove_design
- remove_lib
- which
- search_path

---

read_ddc

Reads Synopsys logical database (.ddc) format design files.

Feedback

Syntax

string *read_ddc*

```
[-netlist_only]
file_names
```

Data Types

```
scenario_name        string
file_names           list
```

Arguments

`-netlist_only`

Reads only the netlist. This option is accepted for backward compatibility but has no effect.

file_names

Specifies a list of files to be read. Typically, only one file is read.

Description

This command reads netlist data from .ddc files into PrimeTime. The .ddc format is a proprietary file format that stores netlist information and constraint data. It is generated by Design Compiler or IC Compiler.

To locate files with relative path names, the command uses the *search_path* variable and searches for each file in each directory listed in the *search_path* variable. Files with absolute path names are loaded as though there were no *search_path* variable.

Examples

The following example reads the mycpu.ddc file based on the *search_path* variable.

```
pt_shell> set search_path "/designs/mycpu/v1.6/dbs /libs/cmos"
/designs/mycpu/v1.6/dbs /libs/cmos

pt_shell> read_ddc mycpu.ddc
Loading ddc file '/designs/mycpu/v1.6/dbs/mycpu.ddc'
Loaded 1 design.
mycpu
1
```

See Also

- read_db
- list_designs

Feedback

- remove_design
- search_path

---

read_dvd

Reads dynamic voltage drop file.

Syntax

status *read_dvd*

```
[-mode 0 | 1]
[-rail_map]
dvd_file
```

Data Types

```
dvd_file     string
```

Arguments

`-mode 0 | 1`

If mode 0 is specified, *read_dvd* parses the effective VDD information from the file. If mode 1 is specified, *read_dvd* parses both power and ground voltages from the dynamic voltage drop file. Default is 0.

`-rail_map`

Provide a means of DvD voltage adjustment for read_dvd timing analsys to cope with the power rail voltage difference in IR drop analysis and in STA analysis. DvD files are generated under certain voltage in RHSC. Sometimes not all corners of a design will have IR-drop analysis performed, while IR-drop awared STA is still required for them. PrimeTime is enhanced to support scaling and offset for DvD through voltage rail map files, now users can use DvD file generated in one corner and scale/offset it to other corners with voltage map files.

dvd_file

Specifies the name of the dynamic voltage drop file.

Description

The *read_dvd* command reads dynamic voltage drop from a disk file.

To locate the file that has relative pathname, the command uses the *search_path* variable and searches for the file in each directory listed in the *search_path* variable. Files that have absolute pathnames are loaded as though there were no *search_path* variable. To determine the file that the *read_dvd* command loads, use the *which* command.

-mode 0 | 1

In the beginning of the file, a format header, with a beginning '#', is required. The format header should contain "inst_name", voltage information, and one of the power pin information. In mode 0, the voltage information is "eff_vdd". In mode 1, the voltage information is "min_vdd_tw" and "max_vss_tw". The power pin information can be "domain", "pg_arc", or "pg_pin". All the dynamic voltage drop information should follow the order of the format header.

-rail_map

“version: 1.0” specify one in the first line in the adjustment table. “object_type” currently only “supply_net” is accepted. “object_spec” defines the supply net name where the scale and offsets are applied, multiple supply nets is allowed in single object_spec. “scaling” is scaling factor for rail mapping between two voltages, the count of the scaling factors should be 1 or 2. Any other combinations of factor definition triggers error message during read_dvd –rail_map and no factor will be applied. “offset” is offset value for rail mapping , the count of offset value follows the same rule as the scaling section.

read_dvd -rail_map is mutually exclusive with *-mode* options, only one of “-rail_map” or “-mode 0|1” can be used.

The *read_dvd* command requires the *timing_enable_dvd_analysis* variable to be set to true.

Examples

The following example shows a dynamic voltage drop file, test.dvd.

-mode 0 | 1

```
pt_shell> sh cat test.dvd
#eff_vdd min_vdd_tw max_vss_tw domain inst_name
0.6700 0.6800 0.0100 VDD buf1
0.6740 0.6820 0.0080 VDDL levelshifter1
0.5620 0.5870 0.0150 VDD levelshifter1

pt_shell> read_dvd [-mode 0 | 1] ./test.dvd
```

-rail_map

```
pt_shell> sh cat [-rail_map] test.dvd
version: 1.0
object_type: supply_net
object_spec: <supply_net1> [<supply_net2> | … ]
scaling: <scale_data> <scale_clock>
offset: <offset_data> <offset_clock>

pt_shell> read_dvd ./test.dvd
```

Feedback

See Also

- remove_dvd
- report_timing
- report_voltage_robustness
- which
- timing_enable_dvd_analysis

---

read_eco_changes

Reads and runs an ECO change list file written by the *write_changes -format eco* command. The command can also be used to run change list generated by PrimeECO.

Syntax

```
status read_eco_changes
[-design_name design_name]
file_name
```

Data Types

```
design_name    string
file_name      string
```

Arguments

`-design_name design_name`

Specifies the name of a lower-level design (block) in the current design or the reference name for a multiply instantiated module (MIM) in the current design. The command applies the changes only to the specified design or MIM group. The *file_name* argument of the command must specify a change list file generated for the design or MIM group.

`file_name`

Specifies the change list file to run.

Description

The *read_eco_changes* command reads an ECO change list file generated by the *write_changes -format eco* command and runs the commands in the file to replay all ECO changes, including both logical and physical changes.

The *read_eco_changes* command can also read a change list file generated by the *implement_eco* command or *write_implement_changes* command in a PrimeECO session and run the logical ECO commands specified in the file.

To read and run block-level design changes from the top-level design, use *-design_name* option.

In the physically aware ECO replay flow, run the *set_eco_options* command to specify the paths to the physical data, such as Design Exchange Format (DEF) and Library Exchange Format (LEF) files. Next, read the physical data by running the *read_eco_changes* command.

LICENSING

When reading and running the ECO commands from a change list generated using a PrimeECO session, the *read_eco_changes* command checks out the *'PrimeECO'* license. In DMSA mode, to utilize the same license for up to 24 scenarios, the user must set the multi-scenario licensing mode to core using the *multi_scenario_license_mode* variable.

Examples

The following example reads and runs the change list files pteco1.data, pteco2.data, and pteco3.data, generated by the *write_changes -format eco* command.

```
pt_shell> read_eco_changes pteco1.data
pt_shell> read_eco_changes pteco2.data
pt_shell> read_eco_changes pteco3.data
```

The following example reads and runs the block-level design change list file, modA.data, generated by the *write_changes -format eco* command for the block-level design "modA" from the top-level design.

```
pt_shell> read_eco_changes -design_name modA modA.data
```

The following example reads and runs the change list files incr2.pt, incr2.pt, and incr3.pt, generated over multiple iterations by *implement_eco* command.

```
pt_shell> read_eco_changes incr1.pt
pt_shell> read_eco_changes incr2.pt
pt_shell> read_eco_changes incr3.pt
```

The following example reads and runs the block-level design change list file, incrA.pt, generated by the *write_implement_changes -format block2top* command for the block-level design "modA" from the top-level design.

```
pt_shell> read_eco_changes -design_name modA incrA.pt
```

See Also

- create_cell
- set_eco_options
- write_changes

Feedback

- eco_enable_mim
- multi_scenario_license_mode

---

read_eco_design

Reads an ECO design written by the *write_eco_design* command.

Syntax

status *read_eco_design*

```
[dir_name]
[-verbose]
```

Data Types

```
dir_name                string
```

Arguments

dir_name

Specifies a directory name from which an ECO design is read. If an ECO design is created by a single scenario PrimeTime session, it must be loaded by a single scenario PrimeTime session. If an ECO design is created by a manager in a multi scenario PrimeTime session, it must be loaded by a manager in a multi scenario PrimeTime session. An ECO design directory created by a single scenario PrimeTime contains multiple files. An ECO design directory created by a multi scenario PrimeTime contains subdirectories with scenario names.

If this option is not specified, the default name, *eco_design_directory* is used.

`-verbose`

Shows additional information while reading an ECO design.

Description

This command seeks to perform ECO in reduced hardware resource compared to the original testcase. This command reads an ECO design written by the *write_eco_design* command. Since the ECO design is only a smaller subset of the original design, its size is much smaller consuming less memory. After the command finishes, you are ready to run the *update_timing* command and perform your timing, DRC and other fixings and write a change list, which is comparable to the change list produced by the fixings in the original design.

Since an ECO design consumes less memory than the original design, you can read the ECO design in smaller memory slots from computing farms like LSF or put more scenarios in a single machine. For example, suppose your original design requires four 100 GB

machines to perform four scenario ECO in Distributed Multi Scenario Analysis (DMSA) in LSF farm. Suppose also its ECO design reduces memory from 100 GB to 20 GB for each scenario. With the ECO design, now you can perform ECO either in four 20 GB machines from LSF farm or in a single 80 GB machine.

Reading a single scenario ECO design

The *read_eco_design* command reads an ECO design from the specified directory name. The following example reads an ECO design from the default directory, *eco_design_directory*.

```
read_eco_design
```

The example below reads an ECO design from the specified directory, *my_eco_design*.

```
read_eco_design my_eco_design
```

Reading a multi scenario ECO design in a single machine

Use the *read_eco_design* command in a DMSA manager to read a multi scenario ECO design created by the *write_eco_design* command in the same or another DMSA manager. The example below reads an ECO design from the *my_dmsa_eco_design* directory created by the *write_eco_design* command in four scenario DMSA.

```
read_eco_design my_dmsa_eco_design
```

If the *set_host_options* command is not used prior to the *read_eco_design* command, PrimeTime analyzes the directory and automatically launches four processes (the same number as scenarios) in the same host to read an ECO design for each process.

You do not need to use the *set_host_options* and *start_host* commands to configure hosts because the *read_eco_design* command automatically configures it. Thus, unlike other DMSA scripts in which host configuration and launching is required prior to timing analysis or ECO, the *read_eco_design* can be the first command of your DMSA script as it automatically configures DMSA hosts for you in a single machine configuration. See sample scripts in the example section below.

When PrimeTime reads a multi scenario ECO design in a single machine, make sure that total memory of all scenarios does not exceed the memory of the machine. For example, if one scenario takes 20 GB peak memory, and there are total 4 scenarios, you must use a machine that has larger memory than 80 GB.

You can use the *-max_cores* option of the *set_host_options command to limit CPU core usages of the processses. If the machine has total 8 cores, the example below limits core usages to 2 cores for each scenario so that total 8 cores are used to run four scenarios.*

```
set_host_options -max_cores 2 -num_processes 4
  start_host
  read_eco_design my_dmsa_eco_design
```

Examples

This section describes a few examples of *read_eco_design* command usages in both single and multiple scenarios.

Single scenario example scripts

Suppose *my_eco_design* is an ECO design directory created by a single scenario *write_eco_design* command. The following example is a script to read the ECO design and performs timing fixing.

```
################################################
# FILE: run_load_and_eco.tcl
#   Reads an ECO design, performs ECO and write
#   a change list for IC Compiler.
################################################
read_eco_design my_eco_design
fix_eco_timing...
write_changes...
exit
```

Example scripts for multi scenario in a single machine

Suppose *my_dmsa_eco_design* is an ECO design created with four scenarios: scen_1, scen_2, scen_3, and scen_4. The original design consumes 100 GB memory for each scenario, but the ECO design consumes only 20 GB for each scenario.

The example below is a script to read the ECO design in a single machine with the same number of processes as the scenarios.

```
#############################################################
# FILE: run_load_and_eco.dmsa.1machine_4proc.tcl
#   Loads an ECO design with four processes in a single
#   machine.
#############################################################
read_eco_design my_dmsa_eco_design
fix_eco_timing...
```

Example scripts for multi scenario in a computing farm

For the same ECO design used in the example above, the script below reads the ECO design using the *bsub* command in LSF machine farm.

```
#############################################################
# FILE: run_load_and_eco.dmsa.lsf.tcl
#   Loads an ECO design with four processes running in LSF
#   hosts. Uses set_host_options command for host
#   configuration.
#############################################################
set_host_options -num_processes 4 \\
                 -submit_command "bsub -R rusage\[mem=20000\] ..." \\
                 -terminate_command "bkill"
```

```
start_host
read_eco_design my_dmsa_eco_design
fix_eco_timing...
```

write_eco_design, read_eco_design, and ECO in one DMSA script

The example below is a typical script running DMSA timing fixing with the same design used above. It requires four 100 GB machines in LSF to run four scenarios for each process.

```
#####################################################################
# FILE: run_dmsa_eco.tcl
#   Typical script for DMSA ECO running from saved sessions.
#   Runs in four 100 GB machines assuming each scenario takes 100 GB.
#####################################################################
#
# Define four scenarios and their session locations
#
create_scenario -name scen_1 -image scen_1_session
create_scenario -name scen_2 -image scen_2_session
create_scenario -name scen_3 -image scen_3_session
create_scenario -name scen_4 -image scen_4_session

#
# Configure hosts with four processes.
#
set_host_options -num_processes 4 \\
                 -submit_command "bsub -R rusage\[mem=100000\] ..."
start_host
current_session -all

#
# ECO
#
fix_eco_timing...
remote_exec { write_changes... }
exit
```

The script below is revised to use an ECO design that runs in four 20 GB machines in LSF. Once the sessions are restored, PrimeTime creates an ECO design serially in one 100 GB machine, and then reads it in four smaller 20 GB machines to perform timing fixing. Note that, only 100 GB memory is used for total peak memory, while the original design used total 400 GB total peak memory.

```
#####################################################################
# FILE: run_dmsa_reduced_resource_eco.tcl
#   Creates an ECO design in one 100 GB machine from LSF.
#   Loads the ECO design in four 20 GB machines from LSF and
#   performs ECO.
#####################################################################
#
# Define four scenarios and their session locations
```

Feedback

```
#
create_scenario -name scen_1 -image scen_1_session
create_scenario -name scen_2 -image scen_2_session
create_scenario -name scen_3 -image scen_3_session
create_scenario -name scen_4 -image scen_4_session

#
# Configure hosts with one 100 GB machine
#
set_host_options -num_processes 1
                 -submit_command "bsub -R rusage\[mem=100000\] ..."
start_host

#
# Create an ECO design with four scenarios in one process.
#
current_session -all
write_eco_design

#
# Stop hosts and remove exising host options
#
stop_host
remove_host_options

#
# Load the ECO design with four 20 GB processes.
#
set_host_options -num_processes 4 \\
                -submit_command "bsub -R rusage\[mem=20000\] ..."
start_host
read_eco_design

#
# ECO
#
fix_eco_timing...
remote_exec { write_changes... }
exit
```

See Also

- read_eco_design
- eco_save_session_data_type

---

read_edge_annotation

Reads boundary leakage edge annotation information from edge annotation file and annotates the boundary edges for boundary leakage analysis.

Feedback

Syntax

status *read_edge_annotation*

```
[-edge_annotation_files file_fmt]
```

Data Types

file_fmt	list

Arguments

`-edge_annotation_files` *file_fmt*

Specifies the name of the edge annotation file.

Description

PrimePower supports boundary leakage power analysis and reporting through edge back annotation flow. The *read_edge_annotation* command reads boundary leakage edge annotation information from edge annotation file and annotates the boundary edges

Examples

The following example reads boundary leakage edge annotation information from edge annotation file

```
pt_shell> read_edge_annotation -edge_annotation_files { edge_annotation }
```

See Also

- set_eco_options
- check_eco
- read_context_leakage_data
- write_edge_annotation

---

read_file

Reads a netlist or library file. This is a DC Emulation command provided for compatibility with Design Compiler.

Syntax

```
status read_file
  [-format db | edif | vhdl | verilog]
  [-single_file ns1]
  [-names_file ns2]
  [-define ns3]
  file_list
```

Feedback

Data Types

ns1	string
ns2	string
ns3	string
file_list	list

Arguments

`-format db | edif | vhdl | verilog`

Specifies the file format.

`-single_file` *ns1*

Not supported by PrimeTime.

`-names_file` *ns2*

Not supported by PrimeTime.

`-define` *ns3*

Not supported by PrimeTime.

file_list

Specifies a list of files that are to be read.

Description

The *read_file* command reads in one or more netlist or library files. The command exists in PrimeTime for compatibility with Design Compiler. Complete information about the *read_file* command can be found in the Design Compiler documentation. The supported method for reading netlist or library files is to use the following commands: *read_db*, *read_verilog*, *read_edif*, and *read_vhdl*.

See Also

- read_db
- read_verilog

---

read_fsdb

Specifies the switching activity information generated by simulation for use in power calculation.

Syntax

status read_fsdb

```
[-path path]
[-strip_path strip_path]
[-zero_delay]
[-rtl]
[-format verilog | vhdl | systemverilog]
[-time window_list]
[-when condition]
[-version]
[-init_start_time init_time]
file_name
```

Data Types

```
path            string
strip_path      string
window_list     list
condition       string
init_time       double
file_name       string
```

Arguments

`-path path`

Specifies a relative path from the current design to the hierarchical low-level design for which the FSDB file has been created. By default, absolute path names are used. Use this option if the FSDB file refers to an object in a hierarchy. Do not use this option if the FSDB file refers to an absolute path.

`-strip_path strip_path`

Specifies a path prefix that is to be stripped from all the object names read from the FSDB file. This option is applied to strip the testbench and instance path from the FSDB file.

`-zero_delay`

Specifies that the FSDB file comes from a zero delay simulation.

`-rtl`

Specifies the FSDB file comes from an RTL simulation. Typically, this means that most nets in the design are not directly annotated from the FSDB, because combinational nets are not included in the RTL simulation. The *set_rtl_to_gate_name* name mapping command is honored only if this option is set. Also, in the *time_based* power analysis mode, event propagation occurs during power analysis only if the *-rtl* option is set when the *read_fsdb* command was issued.

`-format verilog | vhdl | systemverilog`

Specifies the language format for names in the fsdb. Available options are *verilog* (the default), *vhdl*, and *systemverilog*.

`-time` *`window_list`*

Specifies time windows in which the activities are counted for power calculation. The option accepts an even number list of float values in increasing order. For example, *-time {10 20 50 70}* specifies the time window [10ns, 20ns] and [50ns, 70ns]. If you do not know the end of simulation time, use a negative number in the *-time* option to indicate the end of the simulation. For example, *-time {10 -1}* specifies the activity time window from 10 ns to the end of simulation time in the FSDB file. The default activity time window is the whole simulation time in FSDB file. This option applies to averaged, time-based power analysis and statistical leakage power variation analysis. The time is automatically adjusted by the tool according to the timescale in the FSDB file or the *-waveform_interval* value in the *set_power_analysis_option* command. For example, if the *-waveform_interval* is 10, *-time {13 33}* becomes *-time {10 30}*.

`-when` *`condition`*

Specifies a Boolean condition to determine which simulation times from the FSDB should be considered for power analysis. The Boolean condition is a string using names of individual signals in FSDB (either annotated to design objects or not annotated), and also operators. Similar to the *-time* option, the *-when* option selects portions of a FSDB for analysis. However, with the *-when* option, the times selected are chosen automatically based on logical values in the FSDB. During simulation times when the Boolean condition is *true*, the events and power are included in power analysis up to and including the timestamp when the condition becomes false. For simulation times when the Boolean condition is false, the events and power are excluded from analysis up to and including the timestamp when the condition becomes true. The feature can be used in both *averaged* and *time_based* power analysis modes.

`version`

Reports FSDB file version and latest FSDB file version that is currently supported.

`-init_start_time` *`init_time`*

Puts all nets to soft initialization state using short fsdb which later can be overriden by propagation.Option used along with *-time*. Option accepts number of cycles to propagate wave from fsdb. Works only in time_based power analysis mode.

`file_name`

Specifies the switching activity file name to be read.

Description

The *read_fsdb* command specifies the switching activity file generated by simulation to be used for time-based power calculation.

When a FSDB file comes from a zero delay gate level simulation; most of events happen at clock edges. This leads to a very large unrealistic peak power. With the *-zero_delay* option, power is averaged over each clock cycle. This requires you to specify the Synopsys Design Constraint (SDC) file, at least clocks and transition time at primary inputs have to be defined.

If a FSDB file comes from RTL simulation, the *-rtl* option should be used. In this case, the *read_fsdb* attempts to find the name mapping between the objects in the FSDB file and in the gate level netlist automatically. The name mapping rules can be set by the *set_rtl_to_gate_name* command. During the average power calculation, the *update_power* command also automatically propagates the switching activity for those unannotated nets.

Behavior For Different Power Modes

Because of the way analysis is performed in different power analysis modes, the *read_fsdb* command has different behavior depending on the power analysis mode. To specify the power analysis mode, set the *power_analysis_mode* variable to *averaged*, *time_based*, or *leakage_variation*.

In the *averaged* and *leakage_variation* modes, power analysis is performed using toggle rates. In these modes the input activity file is converted to toggle rates and static probabilities, which are annotated onto the design.

In the *time_based* mode, power analysis is performed for individual events in the activity file. The *read_fsdb* command is a required part of this flow. In this flow, the header of the activity file is read when the *read_fsdb* command is issued, to determine what portion of the design is annotated. The remainder of the activity file is not read until power analysis is performed.

In the *time_based* mode, if you use the *-rtl* option, the tool uses name mapping (see the *set_rtl_to_gate_name* command), and power analysis uses zero delay simulation to create events on unannotated nets. Without the *-rtl* option, name mapping is not used and no events is simulated. It is intended that RTL FSDB be in the input if the option is used. In this case, sequential cells, primary inputs, and black box outputs are typically annotated, while combinational nets are not annotated. Events on these unannotated nets must be produced via simulation.

Examples

The following example reads the FSDB file dump.fsdb from the disk and uses the first 100 ns of the events in the file.

```
pt_shell> read_fsdb dump.fsdb -time {0 100}
```

The following example reads a FSDB file and excludes time periods when the specified condition is false.

```
pt_shell> read_fsdb dump.fsdb -when {net125 && net126}
```

```
pt_shell> read_fsdb dump.fsdb -init_start_time 10 -time {50 100}
```

See Also

- read_vcd
- read_db
- read_verilog
- set_power_analysis_options
- set_rtl_to_gate_name
- power_analysis_mode

---

read_hier_data

Reads binary model information previously written by the *write_hier_data -include model_reload_data* command.

Syntax

string *read_hier_data*

```
        directory
```

Data Types

```
directory     string
```

Arguments

directory

Specifies the directory that contains the binary model.

Description

This command reads in binary model information previously written by the *write_hier_data -include model_reload_data* command. The typical usage of this command is to debug the block model by simulating a top-level run of the block model without really providing a top-level circuit and constraints to instantiate the model. Use this command to directly load the block model into a separate PrimeTime run where the interface circuit of the original block is loaded and timed based on binary data captured in the MODEL directory. Only block models written with reload data are supported by this command.

Feedback

Examples

The following example writes out the binary model info for the current design in the ./ABC directory. The subsequent *read_hier_data* command reads the model information from that directory.

```
pt_shell> write_hier_data ./ABC -include model_reload_data
  ...

pt_shell> ### in a separate PrimeTime session
  ...
pt_shell> read_hier_data ./ABC
  ...
```

See Also

- write_hier_data

---

read_ivm

Reads user-defined via variation tables from a file.

Syntax

status *read_ivm*

```
        interconnect_file
```

Data Types

```
interconnect_file     string
```

Arguments

interconnect_file

Specifies the name of the via interconnect variation file.

Description

The *read_ivm* command reads via interconnect variation tables from a specified text file. Using this data in POCV timing analysis produces more accurate results by taking into account the effects of interconnect via variation.

The following example shows the syntax of the via interconnect variation file.

```
version : 1.0

object_type: design
delay_type: via
```

```
layer_name: *
area            : 0.1
table_min_sigma: 0.25
table_max_sigma: 0.4
table_meanshift: 0
table_stdev     : 0

object_type: design
delay_type: via
layer_name: V1
area            : 0.000400 0.000800  0.001600
table_min_sigma: 0.185     0.143     0.124
table_max_sigma: 0.223     0.204     0.184
table_meanshift: 0         0         0
table_stdev     : 0         0         0

object_type: design
delay_type: via
layer_name: V2
area            : 0.000400 0.000800  0.001600
table_min_sigma: 0.190     0.159     0.128
table_max_sigma: 0.224     0.208     0.192
table_meanshift: 0         0         0
table_stdev     : 0         0         0
```

Currently, only the sigma variation parameters are supported. The meanshift and stdev parameters must be set to zero.

To report the interconnect variation information that has been read in by the *read_ivm* command, use the *report_ivm* command.

Examples

The following command reads in the interconnect variation file named var1.ivm from the current working directory.

```
pt_shell> read_ivm var1.ivm
```

See Also

- remove_ivm
- report_ivm
- timing_enable_via_variation

---

read_lib

Reads in a Synopsys library (.lib) file.

Feedback

Syntax

status *read_lib*

file_name

Data Types

file_name	list

Arguments

file_name

Specifies the name of a library file to read.

Description

The *read_lib* command reads a Synopsys library (.lib) file into the PrimeTime tool. To locate files that have relative path names, the command uses the *search_path* variable and searches for each file in each directory listed in the *search_path* variable. Files that have absolute path names are loaded as though there is no *search_path*. To determine the file that the *read_lib* command loads, use the *which* command. After the library file is loaded, to list the library objects, use the *list_libs* command. To remove libraries, use the *remove_lib* command.

If the *power_enable_analysis* variable is set to *true*, the *read_lib* command also loads power models. If the variable is *false*, neither the *read_lib* nor *link_design* command nor the first power related command loads power models. The incremental power model loading only works on .db files by using either the *link_library* or *read_db* command or both of these commands. In other words, if you use .lib files and you want to do power analysis, you must set the *power_enable_analysis* variable to *true* and use the *read_lib* command to read in your .lib files.

For more information, see the Library Compiler documentation.

Examples

In the following example, the file bus1.lib is read based on the *search_path*.

```
pt_shell> set search_path "/designs/newcpu/v1.6 /libs/cmos"
/designs/newcpu/v1.6 /libs/cmos

pt_shell> read_lib bus1.lib
Beginning read_lib...
Reading '/libs/cmos/bus1.lib' ...
Technology library 'bus1' read successfully
1
pt_shell> list_libs
```

Feedback

```
Library Registry:
    bus1            bus1.lib:bus1

1
```

See Also

- list_libs
- remove_lib
- which
- power_enable_analysis
- search_path

---

read_memory_config

Reads the memory states condition from memory configuration file.

Syntax

status *read_memory_config*

`[config_file]`

Data Types

`string config_file`

Arguments

config_file

Specifies the name of the file that contains the when states for memory cell.

Description

The command reads the specified energy when-condition rules for a spacific lib cell in analysis session.

Examples

The following examples shows read memory configuration file.

```
Sample configuration file

CURRENT_CHARACTERIZATION_METHOD PWL

CELL TZLRSREICYMNU3296X59B512M8SW0A

MODE_NAME PWL_READ
```

```
CONDITIONAL_INPUT = CS & !WE & !PD & SLEEPB
CONDITIONAL_PIN = CK { FALL }
END_MODE

MODE_NAME PWL_WRITE
CONDITIONAL_INPUT = CS & WE & !PD & SLEEPB
CONDITIONAL_PIN = CK { RISE }
END_MODE

END

pt_shell> read_memory_config config_file
```

See Also

- set_emmp_configuration
- estimate_memory_max_power

---

read_memory_configuration

Reads the memory states condition from memory configuration file.

Syntax

status *read_memory_configuration*

`[config_file]`

Data Types

`string config_file`

Arguments

config_file

Specifies the name of the file that contains the when states for memory cell.

Description

The command reads the specified energy when-condition rules for a spacific lib cell in analysis session.

Examples

The following examples shows read memory configuration file.

```
Sample configuration file

CURRENT_CHARACTERIZATION_METHOD PWL

CELL TZLRSREICYMNU3296X59B512M8SW0A
```

```
MODE_NAME PWL_READ
CONDITIONAL_INPUT = CS & !WE & !PD & SLEEPB
CONDITIONAL_PIN = CK { FALL }
END_MODE

MODE_NAME PWL_WRITE
CONDITIONAL_INPUT = CS & WE & !PD & SLEEPB
CONDITIONAL_PIN = CK { RISE }
END_MODE

END

pt_shell> read_memory_configuration config_file
```

See Also

- set_emmp_configuration
- estimate_memory_max_power

---

read_milkyway

Reads in a linked design from a Milkyway database.

Syntax

status *read_milkyway*

```
[-version version]
[-netlist_only]
[-library path]
[-scenario scenario_name]
CEL_name
```

Data Types

version	int
path	string
scenario_name	string
CEL_name	string

Arguments

`-version version`

Specifies the version of the design to be read. For example, there are design files under the CEL view in the Milkyway design library design_lib:

```
\\'design_lib/CEL/design1_pre_route1:1\\'
\\'design_lib/CEL/design1_post_route:2\\'
```

The 1 or 2 after the ':' is the version number of the design. The default is to read the most current version.

`-netlist_only`

Reads only the netlist is to be read; does not read constraints. By default, the command reads both netlist and constraints.

`-library` *`path`*

Specifies the absolute or relative path to the Milkyway design library. You can omit this option if the *mw_design_library* variable specifies the path to the Milkyway design library.

`-scenario` *`scenario_name`*

The Milkyway database can store multiple constraints that can correspond to various scenarios of running the design. This option specifies the name of the scenario for reading in constraints from Milkyway database. The default is to not use a scenario.

`CEL_name`

Specifies the design file name to be read. For example, there are design files under the CEL view in the Milkyway design library design_lib:

```
\\'design_lib/CEL/design1_pre_route1:1\\'
\\'design_lib/CEL/design1_post_route:2\\'
```

The design1_pre_route or design1_post_route are the CEL_name argument. Do not include version number in this argument.

Description

This command reads a linked design from Milkyway database into PrimeTime. The command can also optionally load the timing constraints from the Milkyway database.

The *read_milkyway* command always reads in a linked design. Before you run this command, you need to set the *link_path* and *search_path* variables. If a linked design already exists when you run the *read_milkyway* command, the command unlinks the existing design before reading the Milkyway design.

Do not run the *link_design* command after the *read_milkyway* command. If you do so, the tool could invalidate certain types of cell hierarchy. This method of reading Milkyway designs is consistent with other tools, but it is different from reading Verilog or VHDL designs, which require an explicit *link_design* command.

Feedback

The following variables affect the *read_milkyway* command:

- *link_path* - This variable controls the implicit design linking during the *read_milkyway* command.
- *link_path_per_instance* - If this variable is set so that certain instances have special link paths, the variable setting is honored even in the *read_milkyway* flow.
- *search_path* - This variable is used for automatic loading of the logic libraries.
- *mw_design_library* - If you run the *read_milkyway* command without the *-library* option, the tool gets the library path from this variable. If you specify the *-library* option, this variable is set to the value of the *-library* option.
- *mw_logic0_net* - Use this variable for logic 0 constant nets. The default is "VSS".
- *mw_logic1_net* - Use this variable for logic 1 constant nets. The default is "VDD".

The *link_create_black_boxes* variable does not affect the *read_milkyway* command because the command reads a linked design. If an expected library cell is not present in the libraries specified by the *link_path* or *link_path_per_instance* variable, a black box is created regardless of the *link_create_black_boxes* variable setting.

Examples

In the following example, the latest version of the mw_des design is read into PrimeTime from the mw_db Milkyway database. Any nets set to a constant 0 are created as "VSS", and any set to a constant 1 are created as "VDD".

```
pt_shell> read_milkyway -netlist_only -library ./mw_db mw_des
1
```

After you read a Milkyway database, you can read the parasitics in the database by using the *read_parasitics* command. If a *read_parasitics* command follows the *read_milkyway* command, you do not need to specify a file name for the *read_parasitics* command; it uses the same library and CEL_name as the *read_milkyway* command. For example:

```
pt_shell> read_milkyway -library mw_db mw_des
pt_shell> read_parasitics -format PARA
```

See Also

- link_design
- list_designs
- read_parasitics
- remove_design
- link_path

Feedback

- link_path_per_instance
- mw_design_library
- mw_logic0_net
- mw_logic1_net
- search_path

---

read_ndm

Reads in a netlist from an IC Compiler II design library.

Syntax

status *read_ndm*

```
[-exclude exclude_list]
[-include include_list]
icc2_block_list
```

Data Types

```
exclude_list     list
include_list     list
icc2_block_list  list
```

Arguments

`-exclude exclude_list`

Specifies netlist data to be excluded from reading (which is otherwise included by default):

- *pg_port* - Excludes PG ports

`-include include_list`

Specifies netlist data to be included during reading (which is otherwise excluded by default):

- *physical_only* - Includes physical-only cells as well as functional cells
- *user_pg* - Includes user-defined PG pins

icc2_block_list

Specifies the name of an IC Compiler II block design, or a list of such blocks. The command reads in the netlists for the listed blocks. If a block is hierarchical, the command reads in the netlists of the lower-level blocks as well as the top-level block itself.

Feedback

Specify at least the library name and optionally the path to the library, block name, label name, and view name, using the following form:

```
[path/]library[:block[/label]][.view]
```

For example,

```
my_lib                          reads netlists from all blocks in
 my_lib
my_lib:blockA                   reads netlist from blockA in
 my_lib
my_lib:blockA/postroute         reads netlist from postroute
 blockA
my_lib:blockA/postroute.design  reads netlist from postroute
 blockA design view
```

You can specify a relative or absolute path to the directory containing the library directory, as in the following examples.

```
my_dir/mylib:blockA             path relative to the current
 working directory
/home/data/icc2/mylib:blockA    absolute path
```

If you do not specify a path to the library, the command looks in the paths specified by the *search_path* variable in the order listed in the variable, and uses the first library found.

Description

The *read_ndm* command reads one or more IC Compiler II block netlists into the PrimeTime tool for static timing analysis.

By default, when reading the netlist data, the command includes PG ports, excludes physical-only cells, and excludes user-defined PG pins. You can change the default behavior by using the *-exclude* and *-include* options.

This command reads only netlist data. To read physical design data for the physically aware ECO flow, use the *set_eco_options* command with the *-physical_icc2_lib* and *-physical_icc2_blocks* options

Examples

The following command reads in the netlist from the IC Compiler II block named Top in the MyDesigns library:

```
pt_shell> read_ndm MyDesigns:Top
Loading NDM design 'Top'
1
```

The MyDesigns library must be in the current working directory or in a directory specified by the *search_path* variable.

Feedback

The following command reads in the netlist from the IC Compiler II block named Top, post_route label, in the MyDesigns library in a specific absolute path:

```
pt_shell> read_ndm /remote/icc2data/MyDesigns:Top/post_route
Loading NDM design 'Top'
1
```

The following commands set the *search_path* variable and read in the netlists from two blocks in the MyDesigns library. The command looks for the MyDesigns library directory in the current working directory first and then in the /remote/icc2data directory.

```
pt_shell> set_app_var search_path ". /remote/icc2data"
 . /remote/icc2data
pt_shell> read_ndm {MyDesigns:Top MyDesigns:Blk1}
Loading NDM design 'Top'
Loading NDM design 'Blk1'
1
```

See Also

- list_designs
- read_verilog
- remove_design
- set_eco_options
- search_path

---

read_ndm_lib

Reads in a CLIB library file in the Fusion Compiler flow.

Syntax

status *read_ndm_lib*

```
        clib_file_list
```

Data Types

clib_file_list list

Arguments

clib_file_list

Specifies the name of a CLIB file or a list of CLIB files to read.

If you do not specify a path to the CLIB file, the command looks in the paths specified by the *search_path* variable in the order listed in the variable, and uses the first CLIB library found.

Description

The *read_ndm_lib* command reads one or more CLIB files into the PrimeTime tool for static timing analysis. In the Fusion Compiler ECO flow, CLIB data replaces .db library data.

The command returns 1 for success and 0 for failure. It returns 0 if the list contains any CLIB file that contains only physical data and no timing data.

Examples

The following command reads in the CLIB file named MyLib.nlib, specified as an absolute path:

```
pt_shell> read_ndm_lib /home/data/icc2/MyLib.nlib
1
```

The following command reads in the CLIB file named MyLib.nlib, specified as a path relative to the current working directory:

```
pt_shell> read_ndm_lib my_dir/MyLib.nlib
1
```

See Also

- read_ndm
- search_path

---

read_ocvm

Reads advanced and parametric on-chip variation (AOCV/POCV) derating factor tables.

Syntax

status *read_ocvm*

```
[-quiet]
ocvm_file
```

Data Types

```
ocvm_file     string
```

Arguments

`-quiet`

Suppresses the SEL-004, SEL-005, and AOCVM-006 errors and warnings. If this option is not set, errors and warnings are generated when objects specified in the AOCV tables are not found in the current design or libraries.

`ocvm_file`

Specifies the name of the AOCV/POCV file.

Description

The *read_ocvm* command reads AOCV/POCV derate factor tables from a disk file. The tables are annotated onto one or more design objects. Allowed design object types are hierarchical cells, library cells and designs. The AOCV/POCV data is used to reduce pessimism and improve the accuracy of results.

When named AOCV/POCV table sets are specified on objects in the design using the *set_ocvm_table_group* command, a cell (or net) derives its AOCV/POCV deratings from the table set associated with the hierarchical cell (or design) that (a) fully contains the cell (or net), and (b) is the lowest level (closest ancestor) hierarchical cell (or design) with an associated AOCV table set. If no such hierarchical cell (or design) containing the cell (or net) is found, the unnamed AOCV/POCV table set is used. Note that a net inherits the AOCV/POCV derate set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV/POCV table set), the lowest level hierarchical cell that fully contains the net is not the block. Therefore, the net derating comes from an AOCV/POCV table set that contains the block and fully encloses the net.

When applying AOCV tables on multiple object types that apply to an arc, the tool uses the following priority, in decreasing order of precedence, to determine the table that applies to the arc:

1. Library cell table
2. Hierarchical cell table
3. Design table

Note: Where a named AOCV table set applies to an arc, the precedence rules above apply to tables that belong to the specified set.

If the *timing_pocvm_precedence* variable is set to *lib_cell_in_file* the precedence of applying POCV tables is consistent with the AOCV precedence. By default, a design level POCV coefficient file takes precedence over LVF data available in library.

Tables are processed in the order that they appear in the AOCV/POCV file. The last table of a specific type {object_type, rf_type, delay_type, derate_type} to be defined for an object in the file overwrites previous tables defined for the same object of the same type.

The *read_ocvm* command can read binary and compressed binary AOCV files created using the *write_binary_aocvm* command. No additional arguments are required to read binary or compressed binary AOCV files.

Examples

The following example shows an AOCV derate file, test.aocvm, with a single table annotated on all library cells.

```
pt_shell> sh cat test.aocvm

version:        1.0

object_type:    lib_cell
rf_type:        rise
delay_type:     cell
derate_type:    late
object_spec:    my_lib/*
voltage:        1.02
depth:          1 2 3
distance:       100 1000
table:          1.20 1.10 1.08
                1.22 1.15 1.11

pt_shell> read_ocvm test.aocvm
```

See Also

- get_timing_paths
- remove_ocvm
- report_ocvm
- report_timing
- reset_ocvm_table_group
- set_aocvm_coefficient
- set_ocvm_table_group
- write_binary_aocvm

Feedback

read_parasitics

Reads net parasitics information from an SPEF, DSPF, RSPF, PARA, or Galaxy Parasitics Database (GPD) and uses it to annotate the currently linked design.

Syntax

status *read_parasitics*

```
[-format file_fmt]
[-complete_with completion_type]
[-lumped_cap_only]
[-pin_cap_included]
[-path prefix]
[-coupling_reduction_factor factor]
[-triplet_type ttype]
[-verbose]
[-syntax_only]
[-eco file_name]
[-original_file_name file_name]
[-keep_capacitive_coupling]
[-x_offset xlist]
[-y_offset ylist]
[-axis_flip flip_list]
[-rotation rot_list]
[-parent]
file_names
```

Data Types

completion_type	string
factor	float
file_fmt	string
file_name	string
file_names	string
flip_list	string
prefix	string
rot_list	string
ttype	string
xlist	string
ylist	string

Arguments

`-format file_fmt`

Specifies the format of the parasitics file. The allowed values are SPEF, DSPF, RSPF, PARA (Milkyway), and GPD. If the *-format* option is not specified, the application can determine whether the file is GPD, SPEF, DSPF, RSPF, or a compressed version of those three ASCII formats. However, to read a file in PARA you must specify the *-format PARA* option.

`-complete_with` *completion_type*

This option does not apply to the RSPF format.

This option indicates that a net with partially annotated parasitics is to be completed by inserting capacitances and resistances according to the *completion_type* argument. The allowed values are *zero*, which completes the net by inserting zero capacitances and resistances, and *wlm*, which completes the net by inserting capacitances and resistances derived from wire load models.

This option is equivalent to reading the parasitics file and then using the *complete_net_parasitics -complete_with* command.

In a hierarchical flow where you read in multiple parasitics files, do not use this option until all parasitics files are read in, or you will create incorrect data for temporarily incomplete networks at block boundaries. Either use *-complete_with* on the final *read_parasitics* command only, or issue an explicit *complete_net_parasitics* command after issuing all *read_parasitics* commands.

Note: The *complete_net_parasitics* and *read_parasitics -complete_with* commands complete a net only if all missing segments are between two pins and the nets are partially annotated (nets are not affected if they are fully annotated or have no annotation at all). The net must also be hierarchical, so that if the parasitics for the block-level parts of a net are missing, those parasitics could exist in the top-level net. If any of these conditions are not met, you must manually correct the SPEF or DSPF file.

`-lumped_cap_only`

This option does not apply to the GPD format.

This option indicates that only the total capacitance of nets is to be annotated as a lumped capacitance on the annotated nets. The RC networks specified in the parasitics file are discarded. The annotated lumped capacitance is the capacitance specified when the net is declared in the parasitics file.

`-pin_cap_included`

Indicates that the RC networks are to include the pin capacitances. By default, the RC network does not include pin capacitances. This option does not apply to the RSPF format. The RC pi model in RSPF format has to always include effect of pin capacitances.

`-path` *prefix*

Specifies a list of relative paths from the current design to the hierarchical instance names for which the parasitics file has been created. By default, absolute path names are used. Use this option if the parasitics file refers to an object (for example, *net*) in a hierarchy (for example, *hier*). Do not use this

option if the parasitics file refers to an absolute path (for example, hier/net). If more than one prefix is supplied, all of the instances of the design can be annotated simultaneously.

`-coupling_reduction_factor` *`factor`*

This option applies only to the SPEF format format. A positive floating point number that specifies the factor to apply when reducing coupling capacitances to grounded capacitances. The default is 1.0. This option is disabled if the *-keep_capacitive_coupling* option is specified. The command fails if both options are specified.

`-triplet_type` *`ttype`*

This option applies only to the SPEF and PARA formats. Capacitor and resistor values in the SPEF and PARA files can be specified as triplets with minimum, typical, and maximum values. Be default, PrimeTime uses the maximum values. Use this option to specify which values are used by PrimeTime: *min*, *typ*, or *max*. The default is *max*.

`-verbose`

Indicates that the *report_annotated_parasitics* report is to be generated when the parasitic file has been read. By default, after reading the parasitics file, the *report_annotated_parasitics* command is not executed.

`-syntax_only`

Indicates that the *read_parasitics* command is to parse the file for syntax errors without performing any parasitic annotation. Use this option to troubleshoot your parasitics file and avoid generating error messages during the actual annotation. No design is required to use the *-syntax_only* option.

`-eco` *`file_name`*

This option specifies the ECO (incremental) parasitic file to be read along with the original parasitic file. The ECO parasitic annotation will update the original annotations. Parasitic errors and warnings for the original parasitic file will be suppressed, and only errors and warnings related to the ECO file will be issued. User should refer to the initial *read_parasitics* command to check for all warnings and errors. Note that in HyperScale top, this command cannot be used to update the parasitic annotations of a block.

This option is presently also supporting the old behavior which indicates that the file being currently annotated is an ECO parasitics from StarRC. PrimeTime SI can read ECO parasitics that are written out only by StarRC. The ECO parasitics can be annotated only when there are some existing parasitics that are already annotated. ECO parasitic files contain re-extracted parasitics for only the ECO nets and their immediate coupling neighbors and do not contain all the nets of

the design. Incremental analysis can be performed after reading ECO parasitics. The Galaxy Incremental Flow supports both the SPEF and GPD formats.

`-original_file_name` *`file_name`*

This option can only be used when the *-eco* option is being used. If the original annotation is performed through multiple parasitic files into PrimeTime SI, the ECO parasitic file corresponds to one of the original files (because it corresponds to one extracted database in StarRC). PrimeTime SI attempts to determine the corresponding original file, but it is not always possible. You can use this option to specify which original parasitic file the ECO file correspond to.

`-keep_capacitive_coupling`

Indicates that the cross capacitors are to be kept in the RC networks data structure. This facilitates the capacitive crosstalk analysis, but does not turn it on. This option disables the *-coupling_reduction_factor* option. The command fails if both options are specified. All coupling capacitors are split to ground with a factor of 1.0 if crosstalk analysis is not activated. This option applies to both the SPEF and the GPD format and requires a PrimeTime SI license.

`-x_offset` *`xlist`*

Indicates a list of x-offsets for all instances specified in the *-path* option. The order in which each x-offsets appears in the *-x_offset* option must correspond to the order of each instance in the *-path* option. Specify the x-offsets in nanometers.

`-y_offset` *`ylist`*

Indicates a list of y-offsets for all instances specified in the *-path* option. The order in which each y-offset appears in the *-y_offset* option must correspond to the order of each instance in the *-path* option. Specify the y-offsets in nanometers.

`-axis_flip` *`flip_list`*

Indicates a list of axis flips for all instances specified in the *-path* option. The order in which each axis flip appears in the *-axis_flip* option must correspond to the order of each instance in the *-path* option. The allowed values are *flip_x*, which indicates a flip across the x-axis, *flip_y*, which indicates a flip across the y-axis, *flip_both*, which indicates a flip along both axis and *flip_none*, which indicates no flip.

`-rotation` *`rot_list`*

Indicates a list of rotations for all instances specified in the *-path* option. The order in which each rotation appears in the *-rotation* option must correspond to the order of each instance in the *-path* option. The allowed values are *rotate_none*, *rotate_90*, *rotate_180*, and *rotate_270*, which indicate a counter-

clockwise rotation of 0, 90, 180, and 270 degrees, respectively. The default is *rotate_none*.

`-parent`

Indicates that the locations values provided by *-x_offset*, *-y_offset*, *-rotation*, or *-axis_flip* are in the coordinate system of the immediate parent hierarchy. By default, they are in the coordinate system of the top level.

`file_names`

When the format is SPEF, DSPF, RSPF, or GPD, this option specifies a list of files from which parasitics information is to be read.

Description

The *read_parasitics* command reads parasitic data from a file and annotates that data on the nets of the current design. For best performance, run this command as soon as possible after the *link_design* command.

The command can read data in SPEF, DSPF, RSPF, and Galaxy Parasitics Database (GPD) format. For SPEF, RSPF, and DSPF, the file can be a simple ASCII file, or it can be compressed with gzip. The *read_parasitics* command automatically detects this format information from the file, but you can specify the base format using the *-format* option.

GPD is an efficient format for sharing parasitics among Synopsys applications. You can use the PrimeTime *write_parasitics* command to write a parasitics file in GPD format.

You can specify parasitics in either reduced or detailed form. Reduced parasitics consist of an RC pi model specified for each driver pin of a net. An RC pi model contains two capacitances and one resistance. The first capacitance connects to the net driver pin; the resistance connects to the net driver pin and to a subnode to which the second capacitance connects. Both capacitances connect to the ground. The parasitics file, using the reduced form, also specifies the pin-to-pin net delays. The RC pi model is used to compute the effective capacitance of the nets at each driver of the net. The effective capacitance determines the cell delays and output slews. You can specify reduced parasitics with the SPEF, DSPF, or RSPF format.

Detailed parasitics consist of a network of resistances and capacitances for each net specified in the parasitics file. The capacitances must be with respect to the ground. Coupling capacitances are not supported unless the *-keep_capacitive_coupling* option is specified and you are using the SPEF or GPD format. Resistances connected to the ground are not connected. The detailed RC network must be complete; incomplete RC networks are discarded but the delays are computed using the incomplete subset of resistances and capacitances. You can specify detailed parasitics with the SPEF, GPD, DSPF, or RSPF formats.

The *read_parasitics* command does not overwrite lumped parasitics (set using the *set_load* or *set_resistance* command) if they already exist on the design. Instead,

a warning is issued. If the lumped parasitics are subsequently removed (using the *remove_capacitance* and *remove_resistance* commands) PrimeTime reverts to the reduced or detailed parasitics.

Incremental annotations are supported; if a small number of nets are affected, a full timing update is not required if the design is already timed. The maximum number of affected nets that avoid a full timing update depends on the total design size.

Multiple resistances and capacitances between the same nodes are also accepted. The values of multiple capacitances between a node are added together and multiple resistances between the same two nodes are kept separately and are considered during the delay calculation.

It is most efficient to read the top level last to minimize the amount of stitching that must be performed.

Net and instance pin names in the design must match instance names in the parasitics file. For example, if you create the parasitics file from a design using VHDL naming conventions, the design name must use VHDL naming conventions.

Parasitics of HyperScale instances are automatically read by PrimeTime when these instances are instantiated at top level, and user-specified parasitics reading of these instances is automatically skipped. Note that if the *-path* option in *read_parasitics* includes multiple instances among which some are HyperScale instances and the others are not, the parasitics reading for those non-HyperScale instances are not skipped.

To remove parasitics that were read and annotated with the *read_parasitics* command, use the *remove_annotated_parasitics* command. The *reset_design* command removes all attributes from the current design, including annotated parasitics.

In PrimeTime, the delay calculation algorithms impose no limit on the number of resistances and capacitances that can be annotated onto a single network; however, there are two shell variables you can manipulate to track and filter annotations that might cause unexpectedly large runtime. Use the *parasitics_warning_net_size* variable to specify the threshold number of nodes for which the PARA-003 message is to be issued, indicating that a large amount of annotation has been encountered for a given network. Similarly, use the *parasitics_rejection_net_size* variable to specify when such detailed annotation is to be rejected in favor of lumped annotation, issuing the PARA-004 warning message.

PrimeTime can also load locations information from parasitic files. This capability is controlled by the *read_parasitics_load_locations* variable. By default, this variable is *false*, and no locations are read into PrimeTime. You can set this variable to *true* to load locations.

Hierarchical parasitics with locations require geometric transformations to be associated with the sub-blocks (i.e.; -path option). PrimeTime computes these transformations based on: user defined transformation; StarRC location guidance for SPEF/GPD; Automatic determination of location transformation.

When reading PARA format the parasitics might not be the same as what is specified by the *-triplet_type* option. This happens when the specified type does not exist. In that case, the *read_parasitics* command chooses something according to the following table:

```
Parasitics stored as
  triplet_type      |   typ   min-max    min-max-typ
--------------------+---------------------------------
    min             |   typ     min          min
    typ             |   typ     max          typ
    max             |   typ     max          max
```

PrimeTime can read a multicorner parasitic file and select one corner of interest. You must specify the corner name by using the *parasitic_corner_name* variable, before using the *read_parasitics* command. After the parasitic corner is set, it cannot be changed between subsequent *read_parasitics* commands. With this feature, the parasitics for only one specific corner from the set of corners is read and annotated. All other parasitic commands would operate on this given corner. The feature is supported for both SPEF and GPD.

For GPD format, the log of *read_parasitics* command will show the list of annotated corners, with the first one being the selected corner for PrimeTime commands. The order of the other corners, if any, is not relevant.

If you read multicorner parasitics without setting the *parasitic_corner_name* variable, the first corner is used for the GPD format, and an error is issued for other formats.

The *parasitic_explorer_enable_analysis* mode allows you to read and annotate all corners in a GPD. However, non-explorer commands still operate on the default corner or the corner specified by *parasitic_corner_name*.

Examples

The following example reads the adder.spef parasitics file from the disk and uses it to annotate the RC parasitics on the current design. The parasitics file contains a description of the RC network for nets of a design.

```
pt_shell> read_parasitics adder.spef
```

The following example shows annotating hierarchical parasitics files. The second command reads parasitics information, for instance u1 of design mult16 from the disk file mult16_u1.spef. The parasitics are annotated on the design, MY_DESIGN. The *-verbose* option enables the automatic call to the *report_annotated_parasitics* command, which would report RC networks of the boundary nets of instance u1 that are not complete.

Pin capacitances are included in the annotated parasitics, as specified by the *-pin_cap_included* option. The second command reads another parasitics file in incremental mode, so that parasitics annotated on the boundary nets of instance u1 are not discarded.

Feedback

```
pt_shell> current_design MY_DESIGN
pt_shell> read_parasitics -pin_cap_included -path u1 mult16_u1.spef
pt_shell> read_parasitics -pin_cap_included -verbose mydesign.spef
```

The following example applies hierarchical parasitics files. The top-level design contains two instances each of block-level designs blkA and blkB.

```
pt_shell> read_parasitics blkA.spf -path {blkA_0 blkA_1}
pt_shell> read_parasitics blkB.spf -path {blkB_0 blkB_1}
pt_shell> read_parasitics top_level.spf
```

The following example completes any partially annotated parasitics after reading in hierarchical parasitics.

```
pt_shell> read_parasitics blkA.spf -path blkA
pt_shell> read_parasitics blkB.spf -path blkB
pt_shell> read_parasitics top_level.spf -complete_with zero
```

The following example removes annotated parasitics information from the current design.

```
pt_shell> remove_annotated_parasitics
```

The following example loads locations into PrimeTime memory.

```
pt_shell> set read_parasitics_load_locations true
pt_shell> read_parasitics adder.spef
```

The following example reads the C_2 corner from a multicorner parasitic file.

```
pt_shell> set parasitic_corner_name "C_2"
pt_shell> read_parasitics multi_corner.spef
```

See Also

- complete_net_parasitics
- remove_annotated_parasitics
- report_annotated_parasitics
- reset_design
- write_parasitics
- read_parasitics_load_locations
- si_enable_analysis
- si_filter_per_aggr_noise_peak_ratio
- parasitic_corner_name
- parasitic_explorer_enable_analysis

Feedback

- PARA-003
- PARA-004

---

read_saif

Reads a Switching Activity Interchange Format (SAIF) file and annotates switching activity information on nets, pins, ports, and cells in the current design.

Syntax

status *read_saif*

```
file_name
[-strip_path prefix]
[-report_inconsistent_annotation report_file]
[-path prefix]
[-ignore ignore_name]
[-exclude exclude_file_name]
[-derate_glitch value]
[-conflict]
[-quiet]
[-parallel strip strip_path inst_name file_name ... ]
[-exclude_hier_pins]
```

Data Types

```
file_name              string
prefix                 string
report_file            string
ignore_name            string
exclude_file_name      string
value                  float
```

Arguments

file_name

Specifies the name of the SAIF file to read. If the file name ends with .gz, .Z, or .bz2, it is read as a gzipped file, a compressed file, or a bzip2ed file, respectively. Otherwise, it is assumed that the file is uncompressed.

-strip_path *prefix*

The *read_saif* command assumes that the current design is instantiated as an instance in the testbench used to generate the SAIF file. The current design, therefore, appears as an instance in the SAIF file. The *-strip_path* argument specifies the name of the instance of the current design as it appears in the SAIF file.

The *read_saif* command annotates all subinstances in the hierarchy of the specified instance and annotates the instance itself. Enter each instance name completely, without any trailing hierarchy separator (/). For example, if the current design is instantiated in the simulation environment as TOP/DUT/U1, use -strip_path TOP/DUT/U1 and not -strip_path TOP/DUT/U.

`-report_inconsistent_annotation` *report_file*

This option specifies the report file name containing objects whose activity is overwritten with different values. When specified, read_saif checks the consistency of the annotated values on objects whose activity values are overwritten, and outputs any inconconsistencies in the report_file. If none occur, the report file is empty.

`-path` *prefix*

Specifies a relative path from the current design to the hierarchical low-level design for which the SAIF file has been created. By default, absolute path names are used. Use this option if the SAIF file refers to an object in a hierarchy. If you want to read the switching information for a portion of the design, you can also use this option. For example, if the SAIF was generated for the whole design and you want to put only switching information on the nets/ports on the boundary and hierarchically under instance add14, then use the *-path add14* option.

`-ignore` *ignore_name*

Specifies the name of an instance in the SAIF file for which switching activity is to be ignored. The *read_saif* command ignores switching activity within the hierarchy of that instance in the SAIF file. *ignore_name* can be a hierarchical name separated by a slash (/). The *-strip_path* option is applied first. The *read_saif* command then strips off the prefix of a net name that matches *prefix* before deciding whether this net name is under the hierarchy specified by the *-ignore* option.

`-exclude` *exclude_file_name*

Specifies the name of a file that contains a list of names to be ignored. The *exclude_file_name* option is used when you want to ignore switching activity for several instances. The file must contain each *ignore_name* on a separate line, without the *-exclude* option. As with the *-ignore* option, the ignore names are recognized after the *-strip_path* argument.

`-derate_glitch` *value*

Specifies a default derating factor value to be used for inertial glitches in the SAIF file. If the *-derate_glitch* option is not specified, a default derating factor of 0.5 is used.

`-conflict`

Indicates that the SAIF file contains conflict activities. These activities will be annotated to conflict cell instances which are specified by *set_power_analysis_options -conflict_activity_cells* and *power_enable_merged_fsdb* variable set to *true* before reading the SAIF file.

`-quiet`

Specifies quiet mode and for instance suppresses warnings on design objects in the SAIF file that could not be annotated on the design.

`parallel`

Specifies parallel saif file reading this option takes multiple of 3 arguments they represents strip_path instance_name and saif file. So, if 2 saif file need to be read parallelly then 6 arguments are passed. The variable power_enable_mt_namemapping need to be set true for parallel name mapping reading. otherwise -parallel option fails.

`-exclude_hier_pins`

This change enables skipping of writing activity information of hierarchical pins when *-exclude_hier_pins* switch is used along with *-rtl* option. This option should only be used with *-rtl* option.

Description

The *read_saif* command reads a SAIF file and annotates the switching activity attributes *toggle_rate* and *static_probability* for nets, ports, and pins of the current design. The *update_power* command uses this information to calculate dynamic power values. If a particular object in the SAIF file cannot be found in the current design, the object is ignored and a warning message is provided. The *read_saif* command returns 1 if at least one of the objects in the file is successfully annotated. Otherwise, it returns 0.

To identify the instance (and corresponding subinstances) you want to annotate, use the *-strip_path* option. To annotate switching information about a particular instance, use the *-path* option. To specify a default derating factor other than 0.5, use the *-derate_glitch* option.

If a SAIF file comes from RTL simulation, the *read_saif* command tries to automatically locate the name mapping between the objects in the SAIF file and in the gate level netlist. The name mapping rules can be set by the *set_rtl_to_gate_name* command. During the average power calculation, the *update_power* command also automatically propagates the switching activity for those unannotated nets.

Examples

The following example annotates switching activity for the current design in the pt_shell, for a design instantiated as Test/U1 in the simulation testbench.

Feedback

```
pt_shell> read_saif file -strip_path Test/U1
```

The following examples annotate the instance U10 under current design.

```
pt_shell> read_saif -file -strip_path Test/U1 -path U10
```

The following examples demonstrate annotation of multiple designs; the design "name1" is instantiated as Test1/U1, and design "name2" is instantiated as Test1/U2.

```
pt_shell> current_design name1

pt_shell> read_saif file_1 -strip_path Test1/U1

pt_shell> current_design name2

pt_shell> read_saif file_2 -strip_path Test1/U2
```

The following examples demonstrate conflict annotation of design. The cell instance "U1" is conflict activity cell and conflict.saif has activities of U1 which conflict with top level switching activities which are from toplevel.saif.

```
pt_shell> set_app_var power_enable_merged_fsdb true

pt_shell> set_power_analysis_options -conflict_activity_cells {U1}

pt_shell> read_saif -conflict -strip_path root conflict.saif

pt_shell> read_saif -strip_path root toplevel.saif
```

See Also

- get_switching_activity
- report_power
- report_switching_activity
- reset_switching_activity
- set_rtl_to_gate_name
- set_switching_activity
- update_power
- set_power_analysis_options
- power_enable_merged_fsdb

Feedback

read_sdc

Reads in a script in Synopsys Design Constraints (SDC) format.

Syntax

int *read_sdc*

```
[-echo]
[-syntax_only]
[-version sdc_version]
[-sms_scenarios sms_scenarios_list]
file_name
```

Data Types

sdc_version	string
file_name	string
sms_scenarios_list	collection

Arguments

`-echo`

Indicates that each constraint is to be echoed as it is executed.

`-syntax_only`

Indicates that SDC script is processed only to check the syntax and semantics of the script.

`-version sdc_version`

Specifies the version of SDC to which the file conforms. Allowed values are *1.0*, *1.1*, *1.2*, *1.3*, *1.4*, *1.5*, *1.6*, *1.7*, *1.8*, *1.9*, *2.0*, *2.1* and *latest* (the default).

`-sms_scenarios sms_scenarios_list`

Specifies a collection of SMS scenarios to be used for the commands included in the file. Each command applies the specified SMS scenarios collection according to thier specification. The option is ignored by commands which do not support the -sms_scenarios option. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

file_name

Specifies the name of the file that contains the SDC script to be read.

Description

The *read_sdc* command reads in a script file in Synopsys Design Constraints (SDC) format that contains commands for use by PrimeTime or by Design Compiler in its Tcl mode. SDC is also licensed by external vendors through the Tap-in program. SDC

formatted script files are Tcl scripts that use a subset of the commands supported by PrimeTime and Design Compiler.

By default, the *read_sdc* command executes the constraints as they are read. If there are errors part way through the script, it is possible that some constraints are applied, and some are not. You can use the *-syntax_only* option to simply check the script without applying any constraints. This option also verifies conformance to the specified SDC version. If there are any errors during the checking phase, the result of the *read_sdc* command is 0.

Like the *source* command, the *read_sdc* command is sensitive to variables that control script execution (for example, the *sh_continue_on_error* and *sh_script_stop_severity* variables). The *read_sdc* command uses the *sh_source_uses_search_path* variable to determine if the *search_path* command is used to find the script.

The *read_sdc* command supports several file formats. The file can be a simple ASCII script file, an ASCII script file compressed with gzip, or a Tcl byte code script, created by TclPro Compiler.

Note that SDC does not allow command abbreviation. Also note that the *exit* and *quit* commands do not cause the application to exit, but they do cause the reading of the SDC file to stop.

The latest SDC version supports the commands listed below. Those added for versions 1.3 and later are noted. Some constraints that PrimeTime ignores are not listed.

General Purpose Commands

```
list
expr
set
```

Object Access Functions

```
all_clocks
all_inputs
all_outputs
current_design
current_instance
get_cells
get_clocks
get_libs
get_lib_cells
get_lib_pins
get_nets
get_pins
get_ports
set_hierarchy_separator
```

Basic Timing Assertions

```
create_clock
create_generated_clock          (1.3)
set_clock_gating_check
set_clock_latency
set_clock_transition
set_clock_uncertainty
set_data_check                  (1.4)
set_false_path
set_input_delay
set_max_delay
set_min_delay
set_multicycle_path
set_output_delay
set_propagated_clock
set_min_pulse_width             (2.0)
set_sense                       (2.1)
```

Secondary Assertions

```
set_disable_timing
set_max_time_borrow
set_timing_derate               (1.5)
```

Environment Assertions

```
set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
set_load
set_logic_zero
set_logic_one
set_logic_dc
set_max_area
set_max_capacitance
set_max_fanout
set_max_transition
set_min_capacitance
set_min_fanout
set_operating_conditions
set_port_fanout_number
set_resistance
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group
```

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note* in Documentation on the Web, which is available through SolvNetPlus at the following address:

https://solvnetplus.synopsys.com/

Feedback

Using some of these commands is restricted when reading SDC. In some cases, some command options are not allowed. In some applications, some of the commands are not supported. For example, PrimeTime does not support the *set_logic** commands. These commands are ignored when loaded in with the *read_sdc* command and a message appears (for an example, see the EXAMPLES section). If an unsupported command is located by the *read_sdc* command, it appears as an unknown command along with a message. The same condition exists for command options. If an option is not supported by SDC, a message is issued indicating that condition. However, if the option is unknown, a different message is issued.

Note that although the *create_generated_clock* command is supported (beginning with version 1.3), there is no provision for the *get_generated_clocks* shortcut, which is available in PrimeTime and Design Compiler. Generated clocks are simply clocks with some additional attributes and in SDC, they are retrieved through the *get_clocks* command.

The following features are added for SDC Version 1.5 or later:

- The *set_clock_latency* command with the *-clock* option
- The *set_timing_derate* command with all of the options
- The *set_max_transition* command with the *-clock_path -data_path -rise*, and *-fall* options
- The *set_clock_uncertainty* command with the *-rise*, *fall*, *from*, and *to* options
- The *set_operating_conditions* command with the *-object_list* option
- Synonyms for all *get_object* commands
- The *get_objects -nocase* command support independently with the *-regexp* option
- Support for the *get_objects -regexp* command
- The *get_objects -of_objects* support for the *get_cells get_nets*, and *get_pins* commands.

The following features are added for SDC Version 2.0 or later:

- The *-reference_pin* option for the commands *set_input_delay* and *set_output_delay*. • The *set_min_pulse_width* command with the *-high* and *-low* options.

The following features are added for SDC Version 2.1 or later:

- The *set_sense* command with the *-type*, *-non_unate*, *-positive*, *-negative*, *-stop_propagation*, *-clock_leaf*, *-pulse*, *-clocks* options. • The *-dynamic* option for the command *set_clock_latency*. • The *-dynamic*, *-static*, and *-increment* options for the command *set_timing_derate*. • The *-ignore_clock_latency* option for the command *set_max_delay*. • The *-ignore_clock_latency* option for the command *set_min_delay*.

The following features are removed for SDC Version 2.1 or later, and are not recognized:

- The *set_clock_sense* command. • The *-multiply_by* option for the command *set_driving_cell*.

Examples

The following example reads the SDC script top.sdc.

```
pt_shell> read_sdc top.sdc -version 1.2
Reading SDC version 1.2...
1
```

The following example reads the SDC script top2.sdc, The script contains commands not supported by SDC, and CMD-005 messages are generated for those commands.

```
pt_shell> read_sdc -syntax_only top2.pt -version 1.2
Checking syntax/semantics using SDC version 1.2...
Error: Unknown command 'link_design' (CMD-005)
** Completed syntax/semantic check with 1 error(s) **
0
```

The following example shows the result when an unsupported command is in an SDC file. One SDC message is generated per instance of an unsupported command. At the end of the read, a summary of all unsupported commands in the file is generated. This SDC file was read into PrimeTime.

```
pt_shell> read_sdc t2.sdc -version 1.2
Reading SDC version 1.2...
Warning: Constraint 'set_logic_zero' is not supported by pt_shell.
 (SDC-3)
Warning: Constraint 'set_logic_zero' is not supported by pt_shell.
 (SDC-3)
Warning: Constraint 'set_logic_one' is not supported by pt_shell. (SDC-3)

Summary of unsupported constraints:
Information: Ignored 1 unsupported 'set_logic_one' constraint. (SDC-4)
Information: Ignored 2 unsupported 'set_logic_zero' constraints. (SDC-4)
1
```

The following example is a script that is not SDC-compliant, because it contains unsupported commands or command options.

```
current_design TOP
set_resistance -value 12.2 [get_nets i1t2]
```

The following example shows the output generated by the *read_sdc* command when reading the previous script. In SDC, the *current_design* command can be used only to get the current design; no arguments are allowed. Also, there is no *-value* option for the *set_resistance* command, so an appropriate message is generated.

```
pt_shell> read_sdc t3.tcl -echo -version 1.2
Reading SDC version 1.2...
current_design TOP
Error: extra positional option 'TOP' (CMD-012)
set_resistance -value 12.2 [get_nets i1t2]
Error: unknown option '-value' (CMD-010)
0
```

The following example shows the message generated when an option is supported by the command but not by SDC.

```
Information: The '-value' option for set_resistance is unsupported.
 (CMD-038)
```

See Also

- source
- write_sdc
- search_path
- sh_continue_on_error
- sh_script_stop_severity
- sh_source_uses_search_path

---

read_sdf

Reads leaf cell and net timing information from a file in Standard Delay Format (SDF) and uses that information to annotate the current design.

Syntax

string *read_sdf*

```
[-path path_name]
[-load_delay cell | net]
[-analysis_type single | bc_wc | on_chip_variation]
[-min_file min_fname]
[-max_file max_fname]
[-type sdf_min | sdf_typ | sdf_max]
[-min_type sdf_min | sdf_typ | sdf_max]
[-max_type sdf_min | sdf_typ | sdf_max]
[-cond_use min | max | min_max]
[-syntax_only]
[-strip_path prefix_path]
[-verbose]
[-worst]
file_name
```

Feedback

Data Types

```
path_name       string
min_fname       string
max_fname       string
prefix_path     string
file_name       string
```

Arguments

`-path path_name`

Specifies the path from the current design to the subdesign for which the timing file has been created.

`-load_delay cell | net`

Specifies whether to read cell (the default) or net load delays in the timing file. The load delay is the portion of cell delay arising from the capacitive load of the net driven by the cell.

`-analysis_type single | bc_wc | on_chip_variation`

Use this option only if you have not already set an analysis type with the *set_operating_conditions -analysis_type* command. The available values are *single bc_wc*, and *on_chip_variation*. If you are in min_max mode, the default is the *bc_wc* value. The *single* value indicates that only one operating condition is to be used.

Specifying either the *bc_wc* or *on_chip_variation* values switches to min_max mode and causes both minimum and maximum delays to be read from the SDF file. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). By default, the *-analysis_type bc_wc | on_chip_variation* option reads the *sdf_min* and *sdf_max* delays, respectively. To change this, use the *-min_type* and *-max_type* options.

`-min_file min_fname`

Specifies the file from which minimum delay timing information is read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0. Use this option only if the minimum and maximum delays are in two separate SDF files.

`-max_file max_fname`

Specifies the file from which maximum delay timing information is read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0. Use this option only if the minimum and maximum delays are in two separate SDF files.

`-type sdf_min | sdf_typ | sdf_max`

Specifies the SDF triplet delay value that is read from the SDF file: *sdf_min*, *sdf_typ*, or *sdf_max* (the default). Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max).

Note: If you use the *-type* option while in min/max mode (for example, if you use the *-operating_conditions bc_bw | on_chip_variation* option), a single value is annotated onto both minimum and maximum values of an arc.

`-min_type sdf_min | sdf_typ | sdf_max`

Specifies the SDF triplet delay value that is read from the SDF file for minimum delay: *sdf_min* (the default), *sdf_typ*, or *sdf_max*. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). Use this option only with the *-analysis_type bc_wc | on_chip_variation* option.

`-max_type sdf_min | sdf_typ | sdf_max`

Specifies the SDF triplet delay value that is read from the SDF file for maximum delay: *sdf_min*, *sdf_typ*, or *sdf_max* (the default). Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). Use this option only with the *-analysis_type bc_wc | on_chip_variation* option.

`-cond_use min | max | min_max`

Use this option only if the SDF file includes some conditional delays using the SDF construct COND, and if the Synopsys library in use does not specify conditional delays. The available values are *min*, *max*, *min_max*. The *min* value indicates that the minimum of all conditional delays is used to annotate the corresponding timing arc. The *max* value indicates to use the maximum; The *min_max* value indicates min_max operating conditions. The minimum of all conditional delays is used for the minimum operating condition, and the maximum of all conditional delays is used for the maximum operating condition. You cannot use the *min_max* value with a single operating condition; you must be in the min_max mode.

`-syntax_only`

Verifies the SDF syntax. No timing annotation is performed; only syntax is processed. Use this option to verify that your SDF syntax is correct.

`-strip_path` *`prefix_path`*

Specifies a prefix path that is stripped from all SDF objects. Such a prefix path is usually a result of generating an SDF file for a subdesign and using this subdesign as the current design.

`-verbose`

Use this option to enable execution of the *report_annotated_delay* and *report_annotated_check* commands after reading SDF.

`-worst`

Indicates that the *read_sdf* command annotates the current design only with delays worse than the current annotated delays; applies to annotated net and cell delays and annotated timing checks. The worst delay is defined as the most pessimistic delay. This means PrimeTime annotates the minimum of minima and maximum of maxima values.

`file_name`

Specifies the file from which timing information is read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0.

Description

The *read_sdf* command reads from a disk file leaf cell and net timing information and uses it to annotate the current design. The timing file must be in SDF format v1.0, v2.0, v2.1 or v3.0. The file can be a simple ASCII file, or it can be compressed with gzip. Instance-specific pin-to-pin cell and net delays are read from the SDF file and annotated on the current design. When you specify the *-path* option, the *read_sdf* command annotates the current design with information from a timing file created for an instance of a subdesign of the current design. When you specify a subdesign, you cannot use the net delays to the ports of the subdesign to annotate the current design.

The load delay, also known as extra source gate delay, is that portion of the cell delay caused by the capacitive load of the driven net. Some delay calculators consider the load delay part of the net delay. Others consider the load delay part of the cell delay. By default, the *read_sdf* command assumes the load delay is included in the cell delay in the timing file being read. The *-load_delay* option allows you to inform the *read_sdf* command if your timing file includes the load delay in the net delay rather than in the cell delay.

Setup, hold, recovery, removal, period, width, nochange, and skew timing checks, when present in the timing file, are used to annotate the current design. Prior to SDF version 3.0, the SDF construct for removal is HOLD.

Instance names in the design must match instance names in the timing file. For example, if the timing file is created from a design using VHDL naming conventions, the design must use VHDL naming conventions.

To remove delays read and annotated with the *read_sdf* command, use the *reset_design* or *remove_annotated_delay* command. To remove timing checks annotated with the *read_sdf* command, use the *remove_annotated_check* command.

Only partial support for SDF version 3.0 is available. Supported v 3.0 constructs are: REMOVAL, RECREM, RETAIN, and CONDELSE.

Note that the *read_sdf* command does not support hierarchical SDF annotation. All pins must be specified at the leaf cell level. It is not possible to specify hierarchical pins.

When multiple annotations against the same arc are present in the annotation file, the last annotation will take precedence. If the *-worst* option is specified, each annotation is applied only if it is worse than the existing timing value for that arc.

Also note that if the *timing_use_zero_slew_for_annotated_arcs* variable is set to its default *auto* value and at least 95% of delay arcs are annotated, the pure SDF flow is switched on automatically. This results in faster performance of the *update_timing* command, at the expense of no longer calculating the slews on the load pins of annotated arcs. For more information, see the *timing_use_zero_slew_for_annotated_arcs* man page.

Examples

The following example reads the adder.sdf timing file from the disk and uses it to annotate the timing on the current design. The timing file contains load delays included in the cell delays.

```
pt_shell> read_sdf -load_delay cell  adder.sdf
```

The following example reads timing information for instance u1 of design MULT16 from the mult16_u1.sdf disk file that contains the timing for the instance u1 of design MULT16. The timing is annotated on the design MY_DESIGN. In this case, load delay is included in the net delays.

```
pt_shell> current_design MY_DESIGN
pt_shell> read_sdf -load_delay net -path u1 mult16_u1.sdf
```

The following example reads timing information and annotates the current design with the maximum timing when the timing file has different timing conditions for the same pin pair. The load delay is assumed to be included in the cell delay in this example.

```
pt_shell> remove_annotated_delay
pt_shell> read_sdf -cond_use max fname1.sdf
```

The following example reads minimum and maximum timing information and annotates the current design with delays corresponding to minimum and maximum operating conditions, respectively. When reporting minimum delay, PrimeTime uses delays annotated for the minimum condition. When reporting maximum delay, PrimeTime uses delays annotated for the maximum condition.

```
pt_shell> read_sdf -analysis_type bc_wc fname2.sdf
```

The following example reads minimum and maximum timing information from two separate SDF files and annotates the current design with delays corresponding to minimum and maximum operating conditions, respectively. When reporting minimum delay, PrimeTime uses delays annotated for the minimum condition. When reporting maximum delay, PrimeTime uses delays annotated for the maximum condition.

```
pt_shell> read_sdf -analysis_type bc_wc -min_file foo_min.sdf -max_file
 fname3_max.sdf
```

Feedback

See Also

- remove_annotated_check
- remove_annotated_delay
- report_annotated_check
- report_annotated_delay
- reset_design
- set_annotated_check
- set_annotated_delay
- write_sdf
- timing_use_zero_slew_for_annotated_arcs

---

read_signal_em_rules

Reads electromigration (EM) rules from a file in a supported format.

Syntax

status *read_signal_em_rules*

```
-format rule_format
[rule_file]
[-design2itf_map map_file]
[-clear]
```

Data Types

```
string rule_format
string rule_file
string map_file
```

Arguments

`-format rule_format`

Specifies the format of the electromigration rules. The valid values are itf, ircx, and tcl (ratcl format). The default value is itf.

rule_file

Specifies the name of the file that contains the electromigration rules.

`-design2itf_map` *`map_file`*

Specifies the name of the mapping file that defines the name mapping between the layer names used in design database and those in the rule file.

`-clear`

clear loaded electromigration rules from current PrimePower session to release memory.

Description

The command reads the specified electromigration (EM) rules into the analysis session. When successful, the new rules replace the existing ones. These electromigration rules are used to perform electromigration analysis on the signal network. See the *check_signal_em* man page for further details.

In *design2itf_map* mapping file, the conducting_layers section defines the layer name mapping for metal layers. The via_layers section defines the layer name mapping for via layers. In each section, the layer names used in the design database are listed in the first column, where the layer names used in the TLUPlus file are in the second column. Lines starting with "*" are considered as comments.

When electromigration rules are specified in a itf file, this command parsed in the electromigration rule related information specified at header comment section of the itf file. This electromigration rule portion starts with string "#EM_INFORMATION", ends with string "#END_OF_EM_INFORMATION". The itf file might have multiple electromigration rules specified, for example, average, rms and peak. During static electromigration analysis, only average rule is used. During dynamic electromigration analysis, only average, rms and peak rules are used, and you can specify one or all of the three to be reported. See more details in the *check_signal_em* man page.

Regardless of the electromigration rule input format, the electromigration rules establish the current density limits for the geometries from *read_parasitics* command which reads StarRC SPEF file (with tail comment) or GPD database.

Examples

This example replaces the current electromigration rules with the rules read from *MyFile.itfem*:

```
pr_shell> read_signal_em_rules -format itf MyFile.itfem -design2itf_map
 layer_mapping_file
1
```

See Also

- check_signal_em

Feedback

read_tsv_timing

Consume TSV feedthrough path timing data at the stack(top) level.

Syntax

```
int read_tsv_timing
  [-path instance_name]
  tsv_file_name_list
```

Data Types

```
instance_name           string
tsv_file_name_list      list
```

Arguments

`-path instance_name`

Specifies the instance name of the intermediate die.

`tsv_file_name_list`

Specify a list of tsv path timing data files for that intermediate die.

Description

The *write_tsv_timing* command reads in a list of tsv feedthrough path timing data files in binary format at the stack level. Then it loads and annotates the tsv timing data before update_timing for all corners. The recovered skew for source synchronous stack level timing path is shown in *report_timing*.

Note: report_timing will only show the sssr after you use this command.

Examples

The following example reads a list of tsv feedthrough path timing data files at the stack level for corner 1, 2, and 3.

```
prompt> read_tsv_timing -path MD {tsv_1.dat tsv_2.dat tsv_3.dat}
```

See Also

- write_tsv_timing

read_vcd

Specifies the switching activity information generated by simulation for use in power calculation. Internally, non-VCD format switching activity is converted to VCD.

Feedback

Syntax

status read_vcd

```
[-pipe_exec pipe]
[-path path]
[-strip_path strip_path]
[-zero_delay]
[-rtl]
[-format verilog | vhdl | systemverilog]
[-time window_list]
[-when condition]
[-converter_options option_list]
file_name
```

Data Types

```
pipe            string
path            string
strip_path      string
window_list     list
condition       string
option_list     list
file_name       string
```

Arguments

`-pipe_exec pipe`

Specifies a shell command that is used to generate the VCD file *file_name*. This option invokes the command and directly pipe the output VCD file to PrimePower. In other words, the simulation and power analysis are in parallel run. No VCD disk file is generated at all.

`-path path`

Specifies a relative path from the current design to the hierarchical low-level design for which the VCD file has been created. By default, absolute path names are used. Use this option if the VCD file refers to an object in a hierarchy. Do not use this option if the VCD file refers to an absolute path.

`-strip_path strip_path`

Specifies a path prefix that is to be stripped from all the object names read from the VCD file. This option is applied to strip the testbench and instance path from the VCD file.

`-zero_delay`

Specifies that he VCD file comes from a zero delay simulation.

`-rtl`

Specifies the VCD file comes from an RTL simulation. Typically, this means that most nets in the design are not directly annotated from the VCD, because combinational nets are not included in the RTL simulation. The *set_rtl_to_gate_name* name mapping command is honored only if this option is set. Also, in the *time_based* power analysis mode, event propagation occurs during power analysis only if the *-rtl* option is set when the *read_vcd* command was issued.

`-format verilog | vhdl | systemverilog`

Specifies the language format for names in the VCD. Available options are *verilog* (the default), *vhdl*, and *systemverilog*.

`-time` *`window_list`*

Specifies time windows in which the activities are counted for power calculation. The option accepts an even number list of float values in increasing order. The time unit assume to be nanosecond. For example, *-time {10 20 50 70}* specifies the time window [10ns, 20ns] and [50ns, 70ns]. If you do not know the end of simulation time, use a negative number in the *-time* option to indicate the end of the simulation. For example, *-time {10 -1}* specifies the activity time window from 10 ns to the end of simulation time in the VCD file. The default activity time window is the whole simulation time in VCD file. This option applies to averaged, time-based power analysis and statistical leakage power variation analysis. The time is automatically adjusted by the tool according to the timescale in the VCD file or the *-waveform_interval* value in the *set_power_analysis_option* command. For example, if the *-waveform_interval* is 10, *-time {13 33}* becomes *-time {10 30}*.

`-when` *`condition`*

Specifies a Boolean condition to determine which simulation times from the VCD should be considered for power analysis. The Boolean condition is a string using names of individual signals in VCD (either annotated to design objects or not annotated), and also operators. Similar to the *-time* option, the *-when* option selects portions of a VCD for analysis. However, with the *-when* option, the times selected are chosen automatically based on logical values in the VCD. During simulation times when the Boolean condition is *true*, the events and power are included in power analysis up to and including the timestamp when the condition becomes false. For simulation times when the Boolean condition is false, the events and power are excluded from analysis up to and including the timestamp when the condition becomes true. The feature can be used in both *averaged* and *time_based* power analysis modes.

`-converter_options` *`option_list`*

Specifies options for external VCD converters. If the *-converter_options* is specified with list of options, the specified options will be used in conjunction with the external VCD converters, vpd2vcd or fsdb2vcd.

`file_name`

Specifies the switching activity file name to be read. If the *file_name* ends with the .gz, .Z, .bz or .bz2, .vpd, or .fsdb extension, it is read as a gzipped, compressed, bzip2ed, VPD (VCD Plus), or FSDB file, respectively; otherwise, it is assumed to be in the VCD format.

Description

The *read_vcd* command specifies the switching activity file generated by simulation to be used for time-based power calculation.

To read in a gzipped VCD file, a compressed VCD file, a BZIP2 file, a VPD file, and an FSDB file, your PATH should include gunzip, uncompress, bunzip, vcs, and fsdb2vcd (a utility from FSDB inventor Novas http://www.novas.com). The VPD file is generated by VCS. So you should define the environment variable VCS_HOME to tell the tool where to find the VCS utility. If you need to invoke 64-bit VCS utility, you need to set the environment variable VCS_MODE_FLAG to 64. You can also define the environment variable NOVAS_HOME to tell the tool where to find the fsdb2vcd executable.

The *-pipe_exec* option allows PrimePower to read in a VCD file directly from simulator, thus avoiding the need to generate a large VCD file. The VCD file name in this scenario serves as a pipe name. After simulation, the VCD file does not exist. There is no need to modify the existing testbench. The file name is treated like a normal VCD output.(inserting $dumpfile, $dumpvars, etc.) Nor is there any need to invoke the simulator. PrimePower automatically runs the simulation and directly read in the VCD output from the simulator.

When a VCD file comes from a zero delay gate level simulation; most of events happen at clock edges. This leads to a very large unrealistic peak power. With the *-zero_delay* option, power is averaged over each clock cycle. This requires you to specify the Synopsys Design Constraint (SDC) file, at least clocks and transition time at primary inputs have to be defined.

If a VCD file comes from RTL simulation, the *-rtl* option should be used. In this case, the *read_vcd* attempts to find the name mapping between the objects in the VCD file and in the gate level netlist automatically. The name mapping rules can be set by the *set_rtl_to_gate_name* command. During the average power calculation, the *update_power* command also automatically propagates the switching activity for those unannotated nets.

Behavior For Different Power Modes

Because of the way analysis is performed in different power analysis modes, the *read_vcd* command has different behavior depending on the power analysis mode. To specify the

power analysis mode, set the *power_analysis_mode* variable to *averaged*, *time_based*, or *leakage_variation*.

In the *averaged* and *leakage_variation* modes, power analysis is performed using toggle rates. In these modes the input activity file is converted to toggle rates and static probabilities, which are annotated onto the design.

In the *time_based* mode, power analysis is performed for individual events in the activity file. The *read_vcd* command is a required part of this flow. In this flow, the header of the activity file is read when the *read_vcd* command is issued, to determine what portion of the design is annotated. The remainder of the activity file is not read until power analysis is performed.

In the *time_based* mode, if you use the *-rtl* option, the tool uses name mapping (see the *set_rtl_to_gate_name* command), and power analysis uses zero delay simulation to create events on unannotated nets. Without the *-rtl* option, name mapping is not used and no events is simulated. It is intended that RTLVCD be in the input if the option is used. In this case, sequential cells, primary inputs, and black box outputs are typically annotated, while combinational nets are not annotated. Events on these unannotated nets must be produced via simulation.

Examples

The following example reads the VCD file dump.vcd from the disk and uses the first 100 ns of the events in the file.

```
pt_shell> read_vcd dump.vcd -time {0 100}
```

The following example runs VCS and pipes the VCD file dump.vcd directly to PrimePower:

```
pt_shell> read_vcd -pipe_exec "vcs -R tb.v design.v" dump.vcd
```

The following example reads a VCD file and excludes time periods when the specified condition is false.

```
pt_shell> read_vcd dump.vcd -when {net125 && net126}
```

See Also

- read_db
- read_verilog
- set_power_analysis_options
- set_rtl_to_gate_name
- power_analysis_mode

Feedback

read_verilog

Reads in one or more Verilog files.

Syntax

status *read_verilog*

```
[-hdl_compiler]
file_names
```

Data Types

file_names	list

Arguments

`-hdl_compiler`

Reads the Verilog files with HDL Compiler through the PrimeTime external reader (ptxr).

file_names

Specifies the names the files to be read.

Description

This command reads one or more structural, gate-level Verilog netlists into PrimeTime. To a locate file with a relative path name, the command searches for the file in each directory specified by the *search_path* variable. The command locates a file with an absolute path name without considering the *search_path* variable. To determine the file that the *read_verilog* command loads, use the *which* command.

After the Verilog files are loaded, view the design objects by using the *list_designs* command. To remove designs, use the *remove_design* command.

The Verilog netlist must contain fully-mapped, structural designs. PrimeTime cannot link or perform timing analysis with netlists that are not fully mapped at the gate level. There must be no Verilog high-level constructs in the netlist.

Reading Verilog Files With the Native Verilog Reader

By default, the *read_verilog* command invokes the native Verilog reader. For large netlists, this reader is much faster and more memory-efficient than HDL Compiler, but it can only read structural Verilog constructs. The native Verilog reader automatically reads ASCII files or compressed gzip files. The reader uncompresses gzip files to a temporary file in the directory specified by the *pt_tmp_dir* variable.

Generally, the native Verilog reader does not create unconnected nets. If you need to have these nets in your design, set the *svr_keep_unconnected_nets* variable to *true*.

Feedback

Structural Constructs Supported by the Native Verilog Reader

The native Verilog reader supports the following subset of the Verilog language:

- module, endmodule
- input, output, inout
- wire
- tri, wand, wor - These constructs are supported, but they are considered to be the same as the wire construct -- that is, there is no special significance to these constructs.
- supply0, supply1 - These constructs create wires that are logic 0 and 1, respectively.
- assign
- tran - An exception from the gate instantiation subset, tran is supported to the extent that it relates one wire to another, as with assign. Beyond that, tran has no special significance.

If you enable the Verilog preprocessor by setting the *svr_enable_vpp* variable to *true*, the native Verilog reader supports the following directives:

- `define
- `else
- `endif
- `ifdef
- `include
- `undef

The PrimeTime Verilog reader reads and ignores the following constructs

- All simulation directives such as `timescale and `expand_vectornets
- parameter
- defparam
- specify, endspecify

Like HDL Compiler, the native Verilog reader accepts the *translate_off* and *translate_on* comment directives to bypass unsupported Verilog features.

Feedback

- To turn translation off, use one of these directives:

```
// synopsys translate_off
/* synopsys translate_off */
```

- To turn translation back on, use one of these directives:

```
// synopsys translate_on
/* synopsys translate_on */
```

The following example shows how to bypass an unsupported feature:

```
// synopsys translate_off
nand (n2, a1, s2);
// synopsys translate_on
```

Assign Statements and Synonyms

The native Verilog reader creates synonyms for discarded names in an assign statement. One net name is chosen, and the others assigned to it are synonyms. During back-annotation, an explicitly named net can be found through one of its synonyms. Although the *get_nets* command finds nets using their synonyms, it cannot find them when mixed with any wildcards. For example, given the assign statement of assign n1 = n2; where n1 wins, using the get_nets command finds the real net, as in:

```
pt_shell> get_nets n1
{"n1"}
```

You can find the net using the synonym. Notice that the result is the real net, n1, and not the synonym, n2 (which is not a real net):

```
pt_shell> get_nets n2
{"n1"}
```

You cannot use wildcards with synonyms:

```
pt_shell> get_nets n2*
Warning: No nets matched 'n2*' (SEL-004)
Error: Nothing matched for nets (SEL-005)
```

Limitations of the Native Verilog Reader

The native Verilog reader in PrimeTime does not support the following:

- Nonstructural constructs - For more information, see the "Structural Constructs Supported by the Native Verilog Reader" section.
- Parameters - The parameter and defparam statements are read and ignored.
- Gate instantiations

There is very limited support for global naming, that is, referencing a wire from a different module. For example, global.gnd means wire gnd in the global module. Global references are allowed only in instance connections; they cannot be in any other context. In addition, you must ensure the following:

- Reference is a logic constant.
- Global reference is not bussed.
- Referenced module is defined in the same file as the module that is referencing it.
- Referenced module is defined first.
- The global name is used over the default name, but a local name is used over the global name. For example, if global.gnd and 1'b0 are used, gnd is the wire name; however, if ZERO is assigned to 1'b0, and global.gnd is also used, ZERO is used.

Reading Verilog Files With HDL Compiler

Instead of using the native Verilog reader, you can optionally use the HDL Compiler reader, which supports the complete Verilog language. However, compared to the native Verilog reader, the HDL Compiler reader uses more CPU and memory.

To read Verilog files with HDL Compiler, use the *read_verilog* command with the *-hdl_compiler* option. This option requires an HDL Compiler license.

When invoking the HDL Compiler Verilog reader, the PrimeTime external reader (ptxr) reads the .synopsys_dc.setup files (not .synopsys_pt.setup). These include the system, home, and local setup files. The system setup file is always read. However, you can skip the home and local setup files by defining the *ptxr_setup_file* variable in PrimeTime to reference a user-defined ptxr-specific setup file. This user-defined variable does not exist until you define it.

You can use the *bus_naming_style* variable to control bus naming, but setting this variable in PrimeTime does not affect the reading of Verilog files with HDL Compiler.

When using the *-hdl_compiler* option, you can terminate the *read_verilog* command by pressing Ctrl+C three times. This terminates the read process but it does not terminate PrimeTime. This is especially useful if you need to terminate the reading of a very long netlist file.

Limitations of Using HDL Compiler

Reading with the *-hdl_compiler* option has these limitations:

- Any variables that affect Verilog reading are read from .synopsys_dc.setup files, not from .synopsys_pt.setup. These variables cannot be set from within PrimeTime. You can set the *ptxr_setup_file* variable in PrimeTime to restrict ptxr so that it reads only the system setup file and ignores your home and local setup files.

Feedback

- Some of the error messages that appear from the *read_verilog* process cannot be found using the *man* command from PrimeTime. You can access these message man pages from other Synopsys shells or from the UNIX man command.
- Pragmas are ignored.
- No warnings are issued if unmapped logic or HDL constructs exist in the Verilog netlist.

Examples

In the following example, the file newcpu.v is read based on the *search_path*.

```
pt_shell> set search_path "/designs/newcpu/v1.6 /libs/cmos"
/designs/newcpu/v1.6 /libs/cmos

pt_shell> read_verilog newcpu.v
Loading verilog file '/designs/newcpu/v1.6/newcpu.v'
1
pt_shell> remove_design -all
Removing design newcpu...

pt_shell> read_verilog newcpu.v -hdl_compiler
Beginning read_verilog...
Loading db file '/release/libraries/syn/standard.sldb'
Loading db file '/release/libraries/syn/gtech.db'
Loading verilog file '/designs/newcpu/v1.6/newcpu.v'
Reading in the Synopsys verilog primitives.
/designs/newcpu/v1.6/newcpu.v:
1
```

See Also

- list_designs
- remove_design
- which
- bus_naming_style
- pt_tmp_dir
- ptxr_setup_file
- search_path
- svr_enable_vpp
- svr_keep_unconnected_nets

Feedback

read_xdomain_design

Reads an cross-domain design written by the *write_xdomain_design* command.

Syntax

status *read_xdomain_design*

[*dir_name*]

Data Types

dir_name	string

Arguments

dir_name

Specifies a directory name from which a cross-domain design is read. An cross-domain design directory created by PrimeTime contains multiple files.

If this option is not specified, the default name, *xdomain_design_directory* is used.

Description

This command seeks to perform SMVA in reduced hardware resource compared to the original testcase. This command reads an cross-domain design written by the *write_xdomain_design* command. Since the cross-domain design is only a smaller subset of the original design, its size is much smaller consuming less memory. After the command finishes, you are ready to run the *update_timing* command and perform your timing, which is comparable to the original design.

Since a cross-domain design consumes less memory than the original design, you can read the cross-domain design in smaller memory slots from computing farms like LSF. For example, suppose your original design requires four 100 GB machines to perform SMVA in LSF farm. Suppose also its cross-domain design reduces memory from 100 GB to 20 GB. With the cross-domain design, now you can perform SMVA in a 20 GB machine from LSF farm.

Examples

This section describes a few examples of *read_eco_design* command usages in both single and multiple scenarios.

Example scripts

Suppose *my_xdomain_design* is a cross-domain design directory created by *write_xdomain_design* command. The following example is a script to read the cross-domain design and performs timing analysis.

```
################################################
# FILE: run_load_and_xdomain.tcl
#   Reads a cross-domain design, performs SMVA
################################################
set timing_cross_voltage_domain_analysis_mode full
read_xdomain_design my_xdomain_design
update_timing...
report_timing...
exit
```

See Also

- read_xdomain_design

---

record_training_data

Enables recording of user-scripted ECO cell sizing operations for power recovery, allowing faster power recovery in future sessions using similar design data.

Syntax

```
status record_training_data
```

Arguments

This command has no arguments

Description

The *record_training_data* command enables the PrimeTime tool to learn user-scripted ECO cell sizing operations performed by *size_cell* commands for power recovery. After you run the *record_training_data* command, the tool learns the design context surrounding each cell sized by the *size_cell* command. After performing cell sizing operations, use the *write_training_data* command to save the training data.

In future PrimeTime sessions, you can invoke the same cell sizing operations during ECO power recovery for cells in the same or similar design context. When you use the *fix_eco_power* command, use the *-training_data* option to specify the names of the files previously generated by the *write_training_data* command, or use the *training_data_directory* variable to specify the directory where the training data files can be found.

In this flow, the PrimeTime tool does not check or evaluate the quality or benefit of the *size_cell* commands. It simply records the cell sizing changes and saves the timing and topological conditions surrounding those changes for future use during ECO power recovery.

A PrimeTime-ADV-PLUS license is required to use the *record_training_data* command.

Feedback

Examples

The following example starts power optimization training, performs cell sizing for power recovery, and writes the session training data into a file in a specified absolute path.

```
pt_shell> record_training_data
pt_shell> size_cell U100 ...
  ...
pt_shell> source my_eco_changes.tcl  # Cell sizing for power recovery
  ...
pt_shell> size_cell U200 ...
  ...
pt_shell> write_training_data -output /home/train_dir/my_training.td
```

In a future PrimeTime session using the same design or a similar design, the following command invokes the same cell sizing operations for power optimization using the previously saved training data.

```
pt_shell> fix_eco_power -training_data /home/train_dir/my_training.td ...
```

Alternatively, use the *training_data_directory* variable to specify the directory where the training data files can be found:

```
pt_shell> set_app_var training_data_directory /home/train_dir
/home/train_dir
pt_shell> fix_eco_power ...
```

See Also

- fix_eco_power
- write_training_data

---

redirect

Redirects the output of a command to a file.

Syntax

string *redirect*

```
[-append]
[-tee]
[-file | -variable | -channel]
[-compress]
[-bg]
target
command_string
```

Feedback

Data Types

target	string
command_string	string

Arguments

`-append`

Appends the output to the *target* argument.

`-tee`

Like the UNIX command of the same name, sends output to the current output channel as well as to the *target* argument.

`-file`

Indicates that the *target* argument is a file name, and redirection is to that file. This is the default. It is exclusive of the *-variable* and *-channel* options.

`-variable`

Indicates that the *target* argument is a variable name, and redirection is to that Tcl variable. It is exclusive of the *-file* and *-channel* options.

`-channel`

Indicates that the *target* argument is a Tcl channel, and redirection is to that channel. It is exclusive of the *-variable* and *-file* options.

`-compress`

Compresses when writing to file. If redirecting to a file, this option specifies that the output should be compressed as it is written. The output file is in a format recognizable by "gzip -d". You cannot specify this option with the *-append* option.

`-bg`

Indicates that the redirected command executes in the background and in parallel with other foreground commands downstream. PrimeTime returns immediately to execute the next command after this redirect command, unless the next command is *exit* or *quit*, which blocks the run until all background commands with the *-bg* option are finished. Note the *-bg* option must be used together with the *-file* option and no other options are allowed. Redirect command returns the PID of the background process when this option is specified. If the process cannot be successfully started, an empty string is returned.

target

Indicates the target of the output redirection. This is either a filename, Tcl variable, or Tcl channel depending on the command arguments.

command_string

The command to execute. Intermediate output from this command, as well as the result of the command, is redirected to the *target* argument. The *command_string* option should be rigidly quoted with curly braces.

Description

The *redirect* command performs the same function as the traditional UNIX-style redirection operators > and >>. The *command_string* option must be rigidly quoted (that is, enclosed in curly braces) in order for the operation to succeed.

Output is redirected to a file by default. Output can be redirected to a Tcl variable by using the *-variable* option, or to a Tcl channel by using the *-channel* option.

Output can be sent to the current output device, as well as the redirect target by using the *-tee* option. See the Examples section for an example.

The result of a *redirect* command which does not generate a Tcl error, is the empty string. Screen output occurs only if errors occurred during execution of the *command_string* option (other than opening the redirect file). When errors occur, a summary message is printed. See the Examples section for an example.

Although the result of a successful *redirect* command is the empty string, it is still possible to get and use the result of the command that you redirected. Construct a *set* command in which you set a variable to the result of your command. Then, redirect the *set* command. The variable holds the result of your command. See the Examples section for an example.

The *redirect* command is much more flexible than traditional Unix redirection operators. With the *redirect* command, you can redirect multiple commands or an entire script. See the Examples section for an example of how to construct such a command.

Note that the built-in *puts* Tcl command does not respond to output redirection of any kind. Use the built-in *echo* command instead.

The *-bg* option is introduced to enable parallel processing on multicore hardware for performance reasons. PrimeTime returns immediately to execute the next command after this redirect command, unless the next command is *exit* or *quit*, which blocks the process until all background commands launched with the *-bg* option previously are finished. Note the *-bg* option must be used together with the *-file* option and no other options are allowed.

It is also very important to understand that the *-bg* option for the *redirect* command is targeted for post-update reporting and analysis commands. Any command executed in the background causes changes to timing data and triggers incremental update, which result in severe error and stop of execution. In addition, it is advisable to intentionally block

the foreground process if the command to be executed invalidates the existing timing analysis, such as the *update_timing*, *update_noise*, *remove_design* commands. Refer to the *parallel_execute* man page for more details. Use of the *parallel_execute* command is recommended whenever applicable to perform parallel execution of commands.

Examples

In the following example, the output of the plus procedure is redirected. The echoed string and the result of the plus operation is in the output file. Notice that the result was not echoed to the screen.

```
prompt> proc plus {a b} {echo "In plus" ; return [expr $a + $b]}
prompt> redirect p.out {plus 12 13}
prompt> exec cat p.out
In plus
25
```

In this example, a typo in the command created an error condition. The error message indicates that you can use the *error_info* command to trace the error, but you should first check the output file.

```
prompt> redirect p.out {plus2 12 13}
Error: Errors detected during redirect
        Use error_info for more info. (CMD-013)
prompt> exec cat p.out
Error: unknown command 'plus2' (CMD-005)
```

In this example, we explore the usage of results from redirected commands. Since the result of the *redirect* command for a command which does not generate a Tcl error is the empty string, use the *set* command to trap the result of the command. For example, assume that there is a command to read a file which has a result of "1" if it succeeds, and "0" if it fails. If you redirect only the command, there is no way to know if it succeeded.

```
        redirect p.out { read_a_file "a.txt" }
# Now what?  How can I redirect and use the result?
```

But if you set a variable to the result, then it is possible to use that result in a conditional expression and so forth.

```
        redirect p.out { set rres [read_a_file "a.txt"] }
if { $rres == 1 } {
  echo "Read ok!"
}
```

Feedback

The *redirect* command is not limited to redirection of a single command. You can redirect entire blocks of a script with a single *redirect* command. This example with the *echo* command demonstrates this feature:

```
prompt> redirect p.out {
?         echo -n "Hello "
?         echo "world"
?       }
prompt> exec cat p.out
Hello world
prompt>
```

The *redirect* command allows you to tee output to the previous output device and also to redirect output to a variable. This example with the *echo* command demonstrates these features:

```
prompt> set y "This is "
This is
prompt> redirect -tee x.out {
          echo XXX
          redirect -variable y -append {
            echo YYY
            redirect -tee -variable z {
              echo ZZZ
            }
          }
        }
XXX
prompt> exec cat x.out
XXX
prompt> echo $y
This is YYY
ZZZ

prompt> echo $z
ZZZ
```

See Also

- echo
- error_info
- parallel_execute

---

redo

Reverses the effects of a previous *undo* command.

Feedback

Syntax

```
status redo
[-levels num_levels
[-check_only]
[-silent]
```

Data Types

```
num_levels      integer
```

Arguments

`-levels num_levels`

Redoes the effects of the specified number of undo command levels. The default is 1.

`-check_only`

Causes the command to return the same status and messages it would otherwise return without actually changing anything.

`-silent`

Suppresses messages and Tcl errors.

Description

This command redoes the effects of one or more previous *undo* commands.

The redone commands can be undone using the *undo* command again.

Examples

The following example undoes the result of *size_cell* and then redoes it.

```
pt_shell> size_cell InstSimple/InterNand ND2X4
 ...
pt_shell> undo
 ...
pt_shell> redo
 ...
```

See Also

- undo
- undo_config

Feedback

remote_execute

Builds a buffer of commands and triggers execution of these commands on remote workers.

Syntax

status *remote_execute*

```
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-verbose]
command_string
```

Data Types

pre_command_string	string
post_command_string	string
command_string	string

Arguments

`-pre_commands` *pre_command_string*

Specifies a string of commands to be executed in the worker context before the execution of the default command string. Commands are grouped using the semicolon ";" character.

`-post_commands` *post_command_string*

Specifies a string of commands to be executed in the worker context after the execution of the default command string. Commands are grouped using the ";" character.

`-verbose`

Specifies that the worker output should be piped back to the manager console.

command_string

Specifies a string of commands to be executed in the worker context. To group commands, use the semicolon ";" character.

Description

This command is available only if you invoke the pt_shell with the *-multi_scenario* option.

The *remote_execute* command builds up a buffer of commands and triggers execution of these commands on remote workers. All commands in the buffer are executed in the order in which they enter the buffer. This command can only be executed when you have set the current session. Use curly braces { } to force variables and expressions to be evaluated

in a worker context. Use quotation marks (" ") to force variables and expressions to be evaluated in a manager context.

Multi-Scenario Analysis Mode

In the multi-scenario analysis mode, the very first invocation of the *remote_execute* command for a particular session triggers baseline image generation.

Use the *-pre_commands* and *-post_commands* options to specify commands to be executed before and after the commands being remotely executed.

Examples

In the following examples, the s1 scenario is in focus. In the following example, the *remote_execute* command is used to execute a nonmerged report_timing. The output of the *report_timing* command is found in the $multi_scenario_working_directory/s1/out.log file.

```
pt_shell> remote_execute  {report_timing}

Start of Manager/Worker Task Processing
-----------------------------------
Started  : Command execution in scenario 's1'
Command execution in scenario 's1' : Succeeded
Finished : Command execution in scenario 's1'
-----------------------------------
End of Manager/Worker Task Processing
1
```

In the next example, the situation is as in the first example except that the *-pre_commands* option is used to set a variable before the report.

```
pt_shell> remote_execute  -pre_commands {set
rc_slew_lower_threshold_pct_fall 30.0} {report_timing}

Start of Manager/Worker Task Processing
-----------------------------------
Started  : Command execution in scenario 's1'
Command execution in scenario 's1' : Succeeded
Finished : Command execution in scenario 's1'
-----------------------------------
End of Manager/Worker Task Processing
1
```

See Also

- report_multi_scenario_design

Feedback

remove_annotated_check

Removes annotated timing checks from the design, either on specific cells, between specific pins, or on all cells in the current design.

Syntax

status *remove_annotated_check*

```
[-all]
[-from from_pins]
[-to to_pins]
[-rise]
[-fall]
[-clock clock_check]
[-setup | -recovery]
[-hold | -removal]
[-nochange_low]
[-nochange_high]
[-width]
[object_spec]
```

Data Types

from_pins	list
to_pins	list
clock_check	string
object_spec	list

Arguments

`-all`

Removes all annotated timing checks in the design. This option is exclusive of the *-from*, *-to*, and *object_spec* options. Options that specify the type of the timing check, such as *-setup* and *-hold* are ignored when you use the *-all* option.

`-from` *from_pins*

Specifies a list of leaf cell pins that are the startpoints of the timing arcs for which annotated checks are to be removed. You must use the *-from* option with the *-to* option, and you cannot combine these options with the *object_spec* option.

`-to` *to_pins*

Specifies a list of leaf cell pins that are the endpoints of the timing arcs for which annotated checks are to be removed. You must use the *-from* option with the *-to* option, and you cannot combine these options with the *object_spec* option.

`-rise`

Specifies that the timing check is for the rise transition on the constrained (the data *-to* pin). The *-rise* and *-fall* options are mutually exclusive. If you do not specify either the *-rise* or *-fall* option, both values are removed.

`-fall`

Specifies that the timing check is for the fall transition on the constrained (the data *-to* pin). The *-rise* and *-fall* options are mutually exclusive. If you do not specify either the *-rise* or *-fall* option, both values are removed.

`-clock` *clock_check*

Specifies whether the check is for clock rising or falling. By default, checks for both clock rise and fall are removed. The valid values for the *clock_check* options are *rise* or *fall*.

`-setup`

Removes only setup timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the *-setup* option with the *-recover* option.

`-recovery`

Removes only recovery timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the *-setup* option with the *-recover* option.

`-hold`

Removes only hold timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the *-hold* option with the *-removal* option.

`-removal`

Removes only removal timing check information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the *-hold* option with the *-removal* option.

`-nochange_low`

Removes only nochange timing check information against data low information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed;

however, you cannot specify the *-nochange_low* option with the *-nochange_high* option.

`-nochange_high`

Removes only nochange timing check information against data high information. If no timing check option is specified for removal, all annotated timing checks are removed. You can specify more than one type of timing check to be removed; however, you cannot specify the *-nochange_low* option with the *-nochange_high* option.

`-width`

Removes only minimum pulse width check annotation.

`object_spec`

Removes annotated timing checks from the specified list of leaf cells. You cannot combine this option with the *-from* and *-to* options.

Description

This command removes annotated timing checks from the design, either on specific cells, between specific pins, or on all cells in the current design. Timing checks are annotated either from a Standard Delay Format (SDF) file by using the *read_sdf* or *set_annotated_check* command. Timing check types include setup, hold, recovery, removal, and nochange.

You can use the *remove_annotated_check* command in the following ways:

- You can remove all annotated checks from the entire design by using the *-all* option; in this case, you cannot remove a specific check. The command ignores options that specify the type of the timing check such as, the *-setup* and *-hold* options when you use the *-all* option.
- You can remove annotated checks from a list of cells by using the *object_spec* option. Without specifying any timing check type options, the command removes all annotated checks from each cell in the list. By using options that specify the type of the timing check, such as the *-setup* and *-hold* option you can remove only specific checks. You can use this method to remove all checks of a certain type from the entire design. For an example, see the EXAMPLES section.
- You can remove annotated checks between specific pins by using the *-from* and *-to* options. Each from/to pair must be on the same cell. Without specifying any timing check type options, the command removes all annotated checks from each from/to pair. By using options that specify the type of the timing check,, such as the *-setup* and *-hold* options, you can remove only specific checks.

Examples

The following example removes all annotated cell timing checks from the current design.

Feedback

```
pt_shell> remove_annotated_check -all
1
```

The following example removes only setup checks from all cells in the current design.

```
pt_shell> remove_annotated_check -setup [get_cells *]
1
```

The following example removes checks between pins. It also shows the error condition when the pins are not on the same cell and the warning when there are no annotated checks.

```
pt_shell> remove_annotated_check -from ffb/CP -to ffa/D

Error: Cannot remove annotated check from 'ffb/CP' to 'ffa/D':
       pins are on different cells (PTE-032)
0
pt_shell> remove_annotated_check -from ffa/CP -to ffa/D -setup -hold
1
pt_shell> remove_annotated_check -from ffa/CP -to ffa/D
Warning: No annotated timing checks were removed. (PTE-031)
0
```

See Also

- read_sdf
- report_annotated_check
- report_annotated_delay
- reset_design
- set_annotated_check
- set_annotated_delay

---

remove_annotated_clock_network_power

Removes the annotated power on clock networks.

Syntax

status *remove_annotated_clock_network_power*

```
[-clock domain_clock]
```

Data Types

```
domain_clock     string
```

Feedback

Arguments

`-clock` *domain_clock*

Specifies the clock of the related clock network on which the annotated power values are removed.

Description

The *remove_annotated_clock_network_power* command removes the annotated power information specified by the *set_annotated_clock_network_power* command on the current design. If the *-clock* option is used, only the annotated clock network power for the specific clock domain is removed; otherwise, the annotated power values for all clock domains are removed. If the power is not annotated for the specified clock domain, a PWR-294 error message is issued and the command fails.

Examples

In the following example, the *remove_annotated_clock_network_power* command is used before the *report_power* command and after the *set_annotated_clock_network_power* command. The output from the *report_power* command is not affected.

```
pt_shell> set_annotated_clock_network_power -internal 1.0e-03 -switching
 2.0e-03
pt_shell> remove_annotated_clock_network_power
pt_shell> report_power
```

See Also

- report_power
- set_annotated_clock_network_power

---

remove_annotated_delay

Removes annotated delays from the design, either on specific cells or nets, between specific pins, or all annotated delays in the design.

Syntax

status *remove_annotated_delay*

```
[-all]
[-from from_list]
[-to to_list]
[object_spec]
```

Feedback

Data Types

from_list	list
to_list	list
object_spec	list

Arguments

`-all`

Removes all annotated delays in the design. This option is exclusive of the *-from*, *-to*, and *object_spec* options.

`-from` *from_list*

Specifies a list of pins or ports that are the startpoints of the timing arcs for which annotated delays are removed. You cannot combine this option with the *object_spec* option.

`-to` *to_list*

Specifies a list of pins or ports that are the endpoints of the timing arcs for which annotated delays are removed. You cannot combine this option with the *object_spec* option.

object_spec

Removes all annotated delays from the specified list of leaf cells or nets. You cannot combine this option with the *-from* and *-to* options.

Description

The *remove_annotated_delay* command removes annotated cell and net delays from the current design. Delays are annotated either from an SDF file (using the *read_sdf* command) or by using the *set_annotated_delay* command.

There are several ways to use the *remove_annotated_delay* command.

- You can remove all annotated delays from the entire design using the *-all* option.
- You can remove all annotated delays from a list of cells or nets using the *object_spec* option.
- You can remove annotated delays between specific pins using the *-from* and *-to* options.

Examples

The following example removes all annotated net and cell delays from the current design.

```
pt_shell> remove_annotated_delay -all
1
```

The following example removes annotated delays from specific cells.

```
pt_shell> remove_annotated_delay [get_cells u1*]
1
```

The following example removes annotated delays between pins. Also shown is the warning when there are no annotated delays between the specified pins.

```
pt_shell> remove_annotated_delay -from ffb/Q
1
pt_shell> remove_annotated_delay -from ffb/Q -to u1/A
Warning: No annotated delays from 'ffb/Q' to 'u1/A'. (PTE-030)
0
```

See Also

- read_sdf
- remove_annotated_check
- report_annotated_check
- report_annotated_delay
- reset_design
- set_annotated_check
- set_annotated_delay

---

remove_annotated_parasitics

Removes all annotated parasitics from nets of the current design.

Syntax

status *remove_annotated_parasitics*

```
[-all]
[net_list ]
```

Data Types

net_list	list

Arguments

`-all`

Indicates that all annotated nets in the design are to be removed, which is the default. The *-all* and *net_list* options are mutually exclusive.

net_list

Specifies a list of nets from which annotated parasitics are to be removed. By default, all annotated parasitics are removed from the design. The *-all* and *net_list* options are mutually exclusive.

Description

This command removes all net parasitics annotated on nets of the current design. The annotated parasitics are usually added using the *read_parasitics* command to read a SPEF or standard parasitic format (SPF) file and annotate the parasitics on nets of the current design. The *remove_annotated_parasitics* command also removes the annotated parasitics completed by the *complete_net_parasitics* command.

There are two ways to use the *remove_annotated_parasitics* command.

- You can remove all annotated parasitics from the entire design by using the *-all* option or by executing the command without options.
- You can remove all annotated parasitics from a list of nets using the *net_list* option.

Examples

The following example removes all annotated net parasitics from the current design.

```
pt_shell> remove_annotated_parasitics -all
1
```

The following example removes annotated parasitics from the net "CLK".

```
pt_shell> remove_annotated_parasitics [get_nets CLK]
1
```

See Also

- read_parasitics
- report_annotated_parasitics
- reset_design

---

remove_annotated_power

Removes previously annotated power from unresolved black box cells or leaf cells.

Syntax

status *remove_annotated_power*

```
-all
cell_list
```

Feedback

Data Types

cell_list list

Arguments

-all

Specifies that all annotated powers in the design are to be removed. The *-all* and *cell_list* options are mutually exclusive.

cell_list

Specifies a list of cells from which annotated powers are to be removed. The *-all* and *cell_list* options are mutually exclusive.

Description

The *remove_annotated_power* command removes internal power and leakage power that were previously annotated on cells by using the *set_annotated_power* command. There is no way to remove only internal power or leakage power.

Examples

The following example shows how power is annotated, removed, and reported.

```
pt_shell> set_annotated_power -internal_power 1 -leakage_power 0.01 u0/*
1
pt_shell> report_annotated_power -list_annotated
****************************************
Report : annotated_power
        -list_annotated
****************************************

Annotated cell powers:
------------------------------

1.  u0/u0  (internal: 1  leakage: 0.01)
2.  u0/u1  (internal: 1  leakage: 0.01)
                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
------------------------------+-----------+-----------+-----------+
unresolved black-box cell     |         2 |         1 |         1 |
leaf cell                     |         3 |         1 |         2 |
------------------------------+-----------+-----------+-----------+
                              |         5 |         2 |         3 |

1
pt_shell> remove_annotated_power u0/u0
1
pt_shell> report_annotated_power
****************************************
Report : annotated_power
```

```
****************************************

                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
------------------------------+-----------+-----------+-----------+
unresolved black-box cell     |         2 |         1 |         1 |
leaf cell                     |         3 |         0 |         3 |
------------------------------+-----------+-----------+-----------+
                              |         5 |         1 |         4 |

1
```

See Also

- report_annotated_power
- set_annotated_power

---

remove_annotated_transition

Removes previously-annotated transition times from pins or ports in the current design.

Syntax

status *remove_annotated_transition*

```
-all
pin_or_port_list
```

Data Types

```
pin_or_port_list     list
```

Arguments

`-all`

Removes all annotated transition times in the design. You cannot use this option together with the *pin_list* option.

`pin_or_port_list`

Removes annotated transition times from the specified list of pins or ports. You cannot use this option together with the *-all* option.

Description

The *remove_annotated_transition* command removes transition times previously annotated on pins or ports from the current design using the *set_annotated_transition* command.

To view current settings, use the *report_timing* command.

Feedback

Examples

The following example removes all annotated pin and port transition times from the current design.

```
pt_shell> remove_annotated_transition -all
1
```

The following example removes annotated transition time from the input pin *A* of cell "U1/U2/U3".

```
pt_shell> remove_annotated_transition [get_pins U1/U2/U3/A]
1
```

See Also

- remove_annotated_delay
- report_timing
- reset_design
- set_annotated_delay
- set_annotated_transition

---

remove_aocvm

Removes advanced and parametric on-chip variation (AOCV / POCV) information.

Syntax

status *remove_ocvm*

```
[-coefficient]
[-derate]
[object_list]
```

Data Types

object_list list

Arguments

-coefficient

Removes only the AOCV coefficients. The *-coefficient* and *-derate* options are mutually exclusive.

Feedback

`-derate`

Removes only the AOCV/POCV derating factors. The *-coefficient* and *-derate* options are mutually exclusive.

`object_list`

Specifies the objects from which AOCV/POCV information is removed.

Description

Removes user-specified AOCV coefficients and AOCV/POCV derating factors that were previously set with the *set_aocvm_coefficient* and *read_ocvm* commands, respectively. If the command is specified with no options, all AOCV/POCV data is removed.

To display AOCV/POCV deratings and coefficients, use the *report_ocvm* command.

Examples

The following example removes the AOCV coefficients from the library cell named lib/bufA:

```
pt_shell> remove_ocvm -coefficient lib/bufA
```

The following example removes all AOCV/POCV derating factors that were set using the *read_ocvm* command:

```
pt_shell> remove_ocvm -derate
```

The following example removes all AOCV/POCV derating factors from the current design:

```
pt_shell> remove_ocvm -derate [current_design]
```

See Also

- read_ocvm
- report_ocvm
- set_aocvm_coefficient

---

remove_buffer

Removes specified buffers from the current design.

Syntax

int *remove_buffer*

```
[-all_mim_instances]
cell_list
```

Feedback

Data Types

cell_list list

Arguments

cell_list

Specifies a list of buffer cells to be removed.

-all_mim_instances

Removal of buffer cells in one MIM instance is replicated in all instances of a MIM set.

Description

The *remove_buffer* command is used to remove buffer cells from the netlist. A buffer is one or more cells that conforms to the rules the *insert_buffer* command uses to create buffers. Like all other netlist editing commands, for the *remove_buffer* command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The *remove_buffer* command returns a 1 if successful and a 0 if unsuccessful.

If the input and output net of the buffer has parasitics, they are stitched into the resulting net, by default.

The *remove_buffer* command with *-all_mim_instances"* option applies changes to all the instances of the MIM set.

Suppose a CPU module is instantiated four times in the TOP module as CPU1, CPU2, CPU3 and CPU4.

The following example shows *remove_buffer* command being used in one instance of a MIM set i.e CPU1 with *-all_mim_instances"* option. The changes are replicated to all the other MIM instances of the set i.e CPU2, CPU3, CPU4.

```
pt_shell> remove_buffer CPU1/U3 -all_mim_instances"
Removed buffer 'CPU1/U3'.
Removed buffer 'CPU2/U3'.
Removed buffer 'CPU3/U3'.
Removed buffer 'CPU4/U3'.
1
```

Similar to the *insert_buffer* command, the arguments for the *remove_buffer* command can be grouped. The basic interpretation of the *cell_list* is

```
(one non-inverting buffer | two cascaded inverting buffers)+
```

To determine whether a removable buffer configuration exists, the *remove_buffer* command applies the following rules to the cells in the *cell_list*:

- Each cell must be a buffer as defined by the *insert_buffer* command
- The input pin of each cell must be connected to a net
- For multioutput cells, only one output pin can be connected
- There can be no standalone inverting cells
- The output net cannot be multidriven
- Inverter pairs must be connected together and the intermediate net can be connected only to the two inverters

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
```

Feedback

```
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Examples

The following diagram contains inverters U1 and U2:

```
e1/Z -- old_net -- U1 -- net1 --- U2 -- net2 --- e3/A
```

The following example removes U1 and U2.

```
pt_shell> remove_buffer {U1 U2}
Removed buffer 'U1', 'U2'

1
```

After removing the buffer, the configuration is as follows:

```
e1/Z -- old_net --- e3/A
```

See Also

- insert_buffer
- size_cell
- swap_cell
- write_changes

---

remove_capacitance

Removes capacitance on nets or ports.

Syntax

```
status remove_capacitance
  net_or_port_list
```

Data Types

```
net_or_port_list      list
```

Feedback

Arguments

net_or_port_list

Specifies a list of ports and nets in the current design, whose capacitances are removed.

Description

Specifies that the lumped capacitance annotated on the list of nets or ports must be removed. PrimeTime reverts to using the capacitance from detailed parasitics (set using the *read_parasitics* command), if they exist. If not, the internally estimated net capacitance is used.

Annotated capacitances are also removed on the full design by doing a *reset_design*.

Examples

The following example removes the annotated capacitance on net w23.

```
pt_shell> remove_capacitance [get_nets "w23"]
```

The following example removes the annotated capacitance on all ports that name match "P2*".

```
pt_shell> remove_capacitance [get_ports "P2*"]
```

See Also

- all_outputs
- current_design
- set_load
- report_net
- report_port
- reset_design
- set_drive

---

remove_case_analysis

Removes the case analysis value on input.

Syntax

```
status remove_case_analysis
  [-all]
  port_or_pin_list
```

Feedback

Data Types

port_or_pin_list list

Arguments

-all

Removes all logic constant values from the current design. Using this option overrides any port or pin specifications.

port_or_pin_list

Removes case analysis from the specified ports or pins.

Description

This command removes previously specified entries of case analysis on ports or pins. You must specify either the *port_or_pin_list* or *-all* option.

To specify case analysis, run the *set_case_analysis* command.

Examples

The following example specifies that ports in0 and in2 are set to the constant logic value 0:

```
pt_shell> set_case_analysis 0 {in0 in2}
pt_shell> report_case_analysis

****************************************
Report : case_analysis
 ...
****************************************

Pin name                 Case analysis value
---------------------------------------------
in2                      0
in0                      0
```

The following example removes the case analysis entry on port in2:

```
pt_shell> remove_case_analysis {in2}
pt_shell> report_case_analysis

****************************************
Report : case_analysis
 ...
****************************************

Pin name                 Case analysis value
---------------------------------------------
in0                      0
```

Feedback

See Also

- report_case_analysis
- set_case_analysis

---

remove_case_sequential_propagation

Disables case propagation for sequential cell instances or library cells.

Syntax

```
status remove_case_sequential_propagation
  [-all]
  cellinstance_or_libcell_list
```

Data Types

```
cellinstance_or_libcell_list     list
```

Arguments

`-all`

Disables case propagation on all sequential cell instances or library cells.

cellinstance_or_libcell_list

Disables case propagation on the specified sequential cell instances or library cells.

Description

This command disables previously enabled cell instances or library cells for selective sequential propagation. Selective sequential propagation can be enabled for a supported sequential cell using the *set_case_sequential_propagation* command.

To disable sequential propagation on a set of a set of previously enabled cell instances or library cells for sequential propagation, set the *case_analysis_sequential_propagation* variable to *never*.

Examples

The following example enables sequential propagation on cell FF1, FF2, and FF3 and subsequently disables FF1.

```
pt_shell> set_case_sequential_propagation FF1
pt_shell> set_case_sequential_propagation {FF2 FF3}
pt_shell> report_case_analysis -sequential_propagation
****************************************
Report : case_analysis
        -sequential_propagation
```

```
 ...
****************************************

--------------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 3
--------------------------------------------------------------------
Cell Instance : 'FF1'
Cell Instance : 'FF2'
Cell Instance : 'FF3'
--------------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 0
--------------------------------------------------------------------

pt_shell> remove_case_sequential_propagation FF1
pt_shell> report_case_analysis -sequential_propagation
****************************************
Report : case_analysis
        -sequential_propagation
 ...
****************************************

--------------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 2
--------------------------------------------------------------------
Cell Instance : 'FF2'
Cell Instance : 'FF3'
--------------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 0
--------------------------------------------------------------------
```

The following example enables sequential propagation on all cell instances with the same library cell 'snps_test/D_FF' and 'snps_test/D_FF_R'. Additionally, cell instances FF1 and FF2 are enabled in the same list.

```
pt_shell> set_case_sequential_propagation { snps_test/D_FF
 snps_test/D_FF_R FF1 FF2 }
pt_shell> report_case_analysis -sequential_propagation
****************************************
Report : case_analysis
        -sequential_propagation
 ...
****************************************

--------------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 2
--------------------------------------------------------------------
Cell Instance : 'FF1'
Cell Instance : 'FF2'
--------------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 2
--------------------------------------------------------------------
```

```
Library Cell : 'snps_test/D_FF_R'
Library Cell : 'snps_test/D_FF'
```

The following example disables sequential propagation on all enabled cell instances and library cells.

```
pt_shell> remove_case_sequential_propagation -all
pt_shell> report_case_analysis -sequential_propagation
****************************************
Report : case_analysis
        -sequential_propagation
 ...
****************************************

--------------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 0
--------------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 0
--------------------------------------------------------------------
```

See Also

- report_case_analysis
- set_case_analysis
- set_case_sequential_propagation
- case_analysis_sequential_propagation

---

remove_cell

Removes cells from the current design.

Syntax

int *remove_cell*

```
-all
cell_list
```

Data Types

```
cell_list      list
```

Arguments

`-all`

Indicates that all cells are to be removed from the current design. The *-all* and *cell_list* options are mutually exclusive; you can specify only one.

Note: Using the *-all* option at the top of the design effectively removes the entire design.

cell_list

Specifies a list of cells to be removed from the current design. The *-all* and *cell_list* options are mutually exclusive; you can specify only one.

Description

The *remove_cell* command removes cells from the current design. Like all other netlist editing commands, for the *remove_cell* command to succeed, all of its arguments must succeed. If any arguments fail, the netlist remains unchanged. The *remove_cell* command returns a 1 if successful and a 0 if unsuccessful.

Each cell specified must be in scope; that is, at or below the current instance. The *remove_cell* command can be used on leaf cells or hierarchical cells. Duplicate arguments of the *remove_cell* command are ignored with a warning. As the *remove_cell* command progresses, it echoes the cells passed to it. However, only the cells that were actually deleted are logged to the change list. For more information, see the EXAMPLES section.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Examples

In the following example, the first command shows the cells within u1 and u2. The second command removes what appears to be four cells; in fact, more than four cells are deleted. Because u1 and u2 each contain three cells, a total of eight cells are actually deleted. Note also that the change list has only three entries. Because u1/u2 is contained within u1 and is removed after u1, u1/u2 is dropped from the change list.

```
pt_shell> get_cells {u1/* u2/*}
{"u1/u1", "u1/u2", "u1/u3", "u2/u1", "u2/u2", "u2/u3"}
pt_shell> remove_cell {u1 u1/u2 u2/u3 u2}
Information: Removed cell 'u1'. (NED-015)
Information: Removed cell 'u1/u2'. (NED-015)
Information: Removed cell 'u2/u3'. (NED-015)
Information: Removed cell 'u2'. (NED-015)
1
pt_shell> write_changes
########################################
# Change list, formatted for PrimeTime
########################################
current_instance
remove_cell {u1}
current_instance
current_instance u2
remove_cell {u3}
current_instance
remove_cell {u2}
1
```

Feedback

See Also

- size_cell
- swap_cell
- write_changes

---

remove_clock

Removes one or more clocks from the current design.

Syntax

string *remove_clock*

```
-all
clock_list
```

Data Types

```
clock_list      list
```

Arguments

`-all`

Specifies to remove all clocks in the current design.

clock_list

Specifies a list of collections containing clocks or patterns matching the clock names.

Description

Removes the specified clock objects from the current design. You must specify either the *clock_list* or *-all* options.

If a removed clock is the only member of a path group, the path group is also removed. For information about path groups, refer to the *group_path* command man page. To list all clock sources in the design, use the *report_clock* command.

All arrival and required times specified by the *set_input_delay* and *set_output_delay* commands relative to the clock that is being removed are also removed.

Examples

The following example removes clock CLK1.

```
pt_shell> remove_clock CLK1
```

The following example removes all clocks from the current design.

Feedback

```
pt_shell> remove_clock -all
```

See Also

- create_clock
- current_design
- get_clocks
- group_path
- report_clock
- set_input_delay
- set_output_delay
- reset_design

---

remove_clock_exclusivity

Removes the clock exclusivity setting on cell previously set by the *set_clock_exclusivity* command.

Syntax

status *remove_clock_exclusivity*

```
[-output output_pins]
[-all]
```

Data Types

output_pins	list

Arguments

`-output` *output_pins*

Specifies the output pin of a cell that has been previously set as exclusive by the *set_clock_exclusivity* command.

`-all`

Removes all clock exclusivity settings in the current design previously set by the *set_clock_exclusivity* command.

Feedback

Description

This command removes the exclusivity previously set on an output pin of a cell by the *set_clock_exclusivity* command. In that case, the tool will no longer infer exclusivity of clocks that pass through the cell and reach that output pin.

Examples

The following example removes two points of exclusivity previously set by the *set_clock_exclusivity* command:

```
pt_shell> remove_clock_exclusivity -output {AND37/A AND37/B}
```

The following example removes all the exclusivities from the current design.

```
pt_shell> remove_clock_exclusivity -all
```

See Also

- set_clock_exclusivity
- report_clock
- set_disable_auto_mux_clock_exclusivity
- timing_enable_auto_mux_clock_exclusivity

---

remove_clock_gating_check

Removes clock-gating checks.

Syntax

```
status remove_clock_gating_check
   [-setup]
   [-hold]
   [-rise]
   [-fall]
   [-high]
   [-low]
   [-sms_scenarios sms_scenarios_list]
   [object_list]
```

Data Types

```
object_list       list
```

Feedback

Arguments

`-setup`

Removes the clock-gating constraint on the setup time only. If you do not specify either the *-setup* or *-hold* option, both setup and hold constraints are removed.

`-hold`

Removes the clock-gating constraint on the hold time only. If you do not specify either the *-setup* or *-hold* option, both setup and hold constraints are removed.

`-rise`

Removes the clock-gating constraint on the rising delays only. If you do not specify either the *-rise* or *-fall* option, constraints on both rising and falling delays are removed.

`-fall`

Removes the clock-gating constraint on the falling delays only. If you do not specify either the *-rise* nor *-fall* option, constraints on both rising and falling delays are removed.

`-high`

Removes the high specification from the object list, previously set up by the *set_clock_gating_check* command. This option has to be either high or low.

`-low`

Removes the low specification from the object list, previously set up by the *set_clock_gating_check* command. This option has to be either high or low.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which the removal applies. When this option is not given, the removal applies to all SMS scenarios. This collection is created by the *get_sms_scenarios* command.

`object_list`

Specifies a list of objects in the current design from which to remove the clock gating check. The objects can be clocks, pins, or cells. If you specify a cell, all input pins of that cell are affected. If you do not specify any objects, the clock-gating check is removed from the current design.

Description

The *remove_clock_gating_check* command removes clock gating checks for design objects set by the *set_clock_gating_check* command.

An alternative way to remove information set by the *set_clock_gating_check* command is to use the *reset_design* command.

Feedback

Examples

The following example removes the setup requirement (for rising and falling delays) on all gates in the clock network involved with clock CK1 path.

```
pt_shell> remove_clock_gating_check -setup [get_clocks CK1]
```

The following example removes the hold requirement on the rising delay of gate and1.

```
pt_shell> remove_clock_gating_check -hold -rise [get_cells and1]
```

See Also

- remove_disable_clock_gating_check
- report_constraint
- reset_design
- set_clock_gating_check
- set_disable_clock_gating_check

---

remove_clock_groups

Removes specific exclusive or asynchronous clock groups from the current design.

Syntax

```
status remove_clock_groups
   [-logically_exclusive]
   [-physically_exclusive]
   [-exclusive]
   [-asynchronous]
   [-name name_list]
   [-all]
```

Data Types

name_list list

Arguments

-logically_exclusive

Specifies that the groups set for logically exclusive clocks are to be removed. The *-physically_exclusive*, *-logically_exclusive*, and *-asynchronous* options are mutually exclusive; you must choose only one.

Feedback

`-physically_exclusive`

Specifies that the groups set for physically exclusive clocks are to be removed. The *-physically_exclusive*, *-logically_exclusive*, and *-asynchronous* options are mutually exclusive; you must choose only one.

`-exclusive`

Specifies that the groups set for logically exclusive clocks are to be removed.

`-asynchronous`

Specifies that groups set for asynchronous clocks are to be removed. The *-physically_exclusive*, *-logically_exclusive*, and *-asynchronous* options are mutually exclusive; you must choose only one.

`-name` *`name_list`*

Specifies a list of clock groups to be removed, which matches the groups in the given names. These clock groups are predefined by the *set_clock_groups* command. The *-name* and *-all* options are mutually exclusive.

`-all`

Specifies to remove all groups set for exclusive or asynchronous clocks in the current design. The *-name* and *-all* options are mutually exclusive.

Description

This command removes specific exclusive or asynchronous clock groups from the current design. You must specify either the *-exclusive* or *-asynchronous* option. You must specify either the *name_list* or *-all* option.

To list the existing clock groups, run the *report_clock* command with the *-groups* option.

Examples

The following example removes exclusive clock groups named mux.

```
pt_shell> remove_clock_group -exclusive -name mux
```

The following example removes all asynchronous clock groups from the current design.

```
pt_shell> remove_clock_group -asynchronous -all
```

See Also

- get_clock_relationship
- report_clock
- set_clock_groups

Feedback

remove_clock_jitter

Removes clock jitter information previously set by the *set_clock_jitter* command.

Syntax

```
status remove_clock_jitter
   -clock clock_list
```

Data Types

```
clock_list     list
```

Arguments

`-clock clock_list`

Specifies a list of clocks from which to remove the clock jitter.

Description

This command removes the cycle-to-cycle jitter or the duty-cycle jitter of the clock that was previously set by the *set_clock_jitter* command.

Examples

The following example removes the cycle-to-cycle jitter of 0.5 and duty-cycle jitter of 0.7 specified on the clock mclk

```
pt_shell> set_clock_jitter -clock [get_clocks mclk] -cycle 0.5
 -duty_cycle 0.7
pt_shell> remove_clock_jitter -clock [get_clocks mclk]
```

See Also

- report_clock_jitter
- set_clock_jitter

remove_clock_latency

Removes clock latency information from specified objects.

Syntax

string *remove_clock_latency*

```
[-source]
[-clock clock_list]
[-sms_scenarios sms_scenarios_list]
object_list
```

Feedback

Data Types

clock_list	list
object_list	list
sms_scenarios_list	collection

Arguments

`-source`

Specifies that clock source latency should be removed.

`-clock` *clock_list*

Removes any network latency defined on the pin/port objects in the *object_list* that refers the clocks in the *clock_list* option from the design. If the *-clock* option is supplied when the *object_list* refers to clock objects, a warning is issued that the option is not relevant in this case and the command proceeds as if the *-clock* was not provided. This option does not remove a more general latency setting without any specific clock.

object_list

Provides a list of clocks, ports, or pins.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the clock latency compatible with the specified SMS scenarios. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

Description

Removes clock latency information from specified objects. It removes the user-specified clock network or source latency information from specified objects. If a dynamic component of clock source latency has been specified this is also removed.

Clock Network latency is the time it takes for a clock signal on the design to propagate from the clock definition point to a register clock pin. Clock Source latency (also called insertion delay) is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design.

Clock network and source latency information is set on objects using the *set_clock_latency* command. For more information, see the *set_clock_latency* man page.

To report clock network and source latency information, use the *report_clock* command with the *-skew* option.

Examples

The following example removes clock latency information from the *CLK1* clock.

Feedback

```
pt_shell> remove_clock_latency [get_clocks CLK1]
```

The following example removes clock source latency information from the *CLK1* clock.

```
pt_shell> remove_clock_latency -source [get_clocks CLK1]
```

See Also

- create_clock
- remove_clock
- remove_clock_uncertainty
- report_clock
- set_clock_latency
- set_clock_uncertainty

---

remove_clock_map

Removes the user-specified mapping of clocks across hierarchical design levels.

Syntax

status *remove_clock_map*

```
[-instances inst_cells]
[-all]
clock
```

Data Types

```
inst_cells      list
clock           list
```

Arguments

`-instances` *inst_cells*

Removes the user-specified clock mapping from the specified list of cells. Clock mapping can be instance-specific even when the instances refer to same physical block. This option can only be used at top level with to specify removing mapping for subblock instances. It can be only used to remove the user-specified clock mapping that was previously set with the *set_clock_map -instances* command.

`-all`

Removes all user-specified clock mappings.

clock

Removes the user-specified clock mapping for the specified clock.

Description

This command removes user-specified clock mapping for the specified objects that was previously set by the *set_clock_map* command. Note that this command does not remove the automatic clock mapping. The removal of the mapping is corresponding to the set of the clock map. If the clock mapping is set with the *-instance* options, it should be removed with the *-instance* option.

Examples

The following example sets and removes mapping

```
pt_shell> set_clock_map SYSCLK -block_clock BCLK1
pt_shell> set_clock_map SYSCLK -block_clock BCLK2
pt_shell> remove_clock_map SYSCLK
```

The following example sets the mapping and removes all the mappings for blk_inst1, but fails to remove the clock map on SYSCLK for blk_inst2.

```
pt_shell> set_clock_map SYSCLK -block_clock BCLK_1 -instance blk_inst1
pt_shell> set_clock_map SYSCLK -block_clock BCLK_x -instance blk_inst2
pt_shell> remove_clock_map -instance blk_inst1
pt_shell> remove_clock_map SYSCLK
```

See Also

- set_clock_map
- create_clock
- create_generated_clock
- report_clock
- report_constraint
- set_hier_config
- clock_mapping_violations

---

remove_clock_sense

The *remove_clock_sense* command is deprecated. Use the *remove_sense* command instead.

Feedback

See Also

- remove_sense
- set_sense

---

remove_clock_transition

Removes clock transition time information.

Syntax

string *remove_clock_transition*

clock_list

Data Types

clock_list list

Arguments

clock_list

Specifies a list of clocks for which to remove clock transition time.

Description

Removes clock transition information specified by the *set_clock_transition* command. To list all the set clock transition values, use the *report_clock* command with the *-skew* option.

Examples

The following example removes clock transition information from all clocks in the current design.

```
pt_shell> remove_clock_transition [all_clocks]
```

See Also

- all_clocks
- report_clock
- set_clock_transition
- create_clock
- set_clock_latency
- set_clock_uncertainty

Feedback

remove_clock_uncertainty

Removes clock uncertainty information previously set by the *set_clock_uncertainty* command.

Syntax

string *remove_clock_uncertainty*

```
[object_list
[-from from_clock
   | -rise_from rise_from_clock
   | -fall_from fall_from_clock]
[-to to_clock
   | -rise_to rise_to_clock
   | -fall_to fall_to_clock]]
[-rise]
[-fall]
[-setup]
[-hold]
[-sms_scenarios sms_scenarios_list]
```

Data Types

```
object_list           list
from_clock            list
rise_from_clock       list
fall_from_clock       list
to_clock              list
rise_to_clock         list
fall_to_clock         list
sms_scenarios_list    collection
```

Arguments

object_list

Specifies a list of clocks, ports, or pins from which uncertainty information is removed. You can use either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to* options or the *object_list* option, but you cannot specify both; they are mutually exclusive.

`-from` *from_clock*

This option specifies the source clock for interclock uncertainty. You must specify either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to*, or *object_list* options; you cannot specify both.

`-rise_from` *rise_from_clock*

Same as the *-from* option, but indicates that *uncertainty* applies only to rising edge of the source clock. You can use only one of the *-from*, *-rise_from*, or

-fall_from options. Use the *-rise_from* option instead of the obsolete *-from_edge rise* option.

`-fall_from` *`fall_from_clock`*

Same as the *-from* option, but indicates that *uncertainty* applies only to falling edge of the source clock. You can use only one of the *-from*, *-rise_from*, or *-fall_from* options. Use the *-fall_from* instead of the obsolete *-from_edge fall* option.

`-to` *`to_clock`*

This option specifies the destination clocks for interclock uncertainty. You must specify either the pair of the *-from/-rise_from/-fall_from* and *-to/-rise_to/-fall_to*, or *object_list* options; you cannot specify both.

`-rise_to` *`rise_to_clock`*

Same as the *-to* option, but indicates that *uncertainty* applies only to rising edge of the destination clock. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options. Use the *-rise_to* option instead of the obsolete *-to_edge rise* option.

`-fall_to` *`fall_to_clock`*

Same as the *-to* option, but indicates that *uncertainty* applies only to falling edge of the destination clock. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options. Use the *-fall_to* option instead of the obsolete *-to_edge fall* option.

`-rise`

Specifies that uncertainty is removed for only the rising clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty and is now obsolete. Unless you need this option for backward-compatibility, use the *-rise_to* option instead.

`-fall`

Specifies that uncertainty is removed for only the falling clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty and is now obsolete. Unless you need this option for backward-compatibility, use the *-fall_to* option instead.

`-setup`

Specifies that only setup check uncertainty is removed. By default, both setup and hold check uncertainties are removed.

`-hold`

Specifies that only hold check uncertainty is removed. By default, both setup and hold check uncertainties are removed.

Feedback

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the clock uncertainty compatible with the specified SMS scenarios. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

Description

Removes clock uncertainty information previously set by the *set_clock_uncertainty* command from clocks, ports, pins, or between specified clocks. To display clock uncertainty information, use the *report_clock* command with the *-skew* option.

Examples

The following example removes uncertainty information from a clock named *CLK* and a pin named *clk_buf/Z*.

```
pt_shell> remove_clock_uncertainty [get_clocks CLK]
pt_shell> remove_clock_uncertainty [get_pins clk_buf/Z]
```

The following example removes interclock uncertainties between the *PHI1* and *PHI2* clock domains.

```
pt_shell> remove_clock_uncertainty -from PHI1 -to PHI1
pt_shell> remove_clock_uncertainty -from PHI2 -to PHI2
pt_shell> remove_clock_uncertainty -from PHI1 -to PHI2
pt_shell> remove_clock_uncertainty -from PHI2 -to PHI1
```

See Also

- all_clocks
- create_clock
- report_clock
- set_clock_latency
- set_clock_transition
- set_clock_uncertainty

---

remove_command_hook

Remove hook from command execution

Syntax

string *remove_command_hook* -before *name*

Feedback

-after *name* -replace *name commandName*

```
string name
string commandName
```

Arguments

`-before name`

Remove the before hook named *name*

`-after name`

Remove the after hook named *name*

`-replace name`

Remove the replace hook named *name*

commandName

Command to update

Description

The *remove_command_hook* removes a hook previously added to a command. When a hook is created via the *add_command_hook* the name of the hook is returned. Use that name to remove the hook. The get_command_hooks command can also be used to determine the name of each registered hook.

Examples

```
prompt> remove_command_hook place_opt -before before0
```

See Also

- add_command_hook
- get_current_hook_command
- get_command_hooks

---

remove_connection_class

Removes connection class value from ports.

Syntax

integer *remove_connection_class*

object_list

Feedback

Data Types

```
object_list      list
```

Arguments

object_list

Specifies ports whose connection classes are to be removed.

Description

Removes any values for previously set on port with the *set_connection_class* command.

Examples

The following example removes any existing connection class values for an xyz port name.

```
pt_shell> remove_connection_class xyz
```

See Also

- set_connection_class
- report_constraint

---

remove_context

Deletes timing context information previously created by the *characterize_context* command.

Syntax

string *remove_context*

```
-block block_name
[-top]
[-instances cell_list]
```

Data Types

```
block_name    string
cell_list     list
```

Arguments

-block *block_name*

Specifies the name of the design subblock for which its context to be removed. This argument is required.

Feedback

`-top`

Specifies the previously captured top only context information to be removed.

`-instances` *`cell_list`*

Specifies a list of one or more instance names in the current design, all of which must all be instances of the reference block specified by the *-block block_name* argument. If you do not use the *-instances* option, the command affects all instances of the block named by the *-block* argument.

Description

This command deletes the context information previously captured by the *characterize_context* command for the specified list of instances. When you need to recharacterize the context for some instances, first use the *remove_context* command to remove the previously generated context for those instances.

Examples

The following command deletes the context information for instances I1 and I2.

```
pt_shell> remove_context -instances {I1 I2} -block B
```

See Also

- characterize_context
- write_context
- report_context

---

remove_context_margin

Removes context margin information previously set by the *set_context_margin* command.

Syntax

status *remove_context_margin*

```
[-all]
[-global]
[-type clock | data]
[objects]
```

Data Types

type	string
objects	list

Feedback

Arguments

`-all`

Removes all included context margins inside *object*. If no object is specified, it removes all context margins in the current design.

`-global`

Removes the global context margin in this design.

`-type clock | data`

Removes context margins only for the specified type, either *clock* or *data*. By default, both types are removed.

`objests`

Specifies a list of instances or pins/ports for which the context margin is removed

Description

The *remove_context_margin* command removes context margins previously set by the *set_context_margin* command. You can choose to delete only clock or data margin by specifying the *-type* option.

By default, *remove_context_margin* removes only context margin on the specified *objects*. To delete context margins on all objects, use the *-all* option.

Examples

The following example removes only the data context margin on BLK.

```
pt_shell> remove_context_margin -type data BLK
```

The following example removes all context margins in the current design.

```
pt_shell> remove_context_margin -all
```

The following example removes the clock context margins on BLK and all context margins on boundary pins inside BLK.

```
pt_shell> remove_context_margin -all -type clock BLK
```

See Also

- set_context_margin
- report_context

Feedback

remove_coupling_separation

Removes the constraints set by the *set_coupling_separation* command.

Syntax

```
status remove_coupling_separation
   [-pairwise pnets]
   -all
   nets
```

Data Types

```
pnets            list
nets             list
```

Arguments

`-pairwise` *pnets*

Specifies that all coupling capacitances between only *pnets* and *nets* are excluded. The *-pairwise* option can only be used to remove separation constraints applied on the nets using the *set_coupling_separation* command with the *-pairwise* option. Any coupling separation applied on the nets globally without the *-pairwise* option cannot be reset by this option.

`-all`

Resets coupling separation constraints on all nets.

nets

Specifies a list of nets from which separation constraints are to be removed.

Description

This command removes the constraints set by the *set_coupling_separation* command.

This command restores any prior settings on *pnets* or *nets* from the *set_si_delay_analysis* or *set_si_noise_analysis* command.

Instances of this command are recorded for output by the *write_changes* command. This feature allows you to communicate the separation constraint to other implementation tools along with netlist changes.

To verify the result of this command, run the *report_si_delay_analysis* or *report_si_noise_analysis* command with the *-coupling_separated* option.

Examples

In the following example, all the nets named CLK_NET_* are returned to their original state in crosstalk analysis.

Feedback

```
pt_shell> set_coupling_separation [get_nets CLK_NET*]
1
pt_shell> remove_coupling_separation [get_nets CLK_NET*]
1
```

In the following example, the separation is applied pairwise on the nets and is removed by using the *-pairwise* option.

```
pt_shell> set_coupling_separation -pairwise [get_nets {n1 n2}] [get_nets
 n3]
1
pt_shell> remove_coupling_separation -pairwise [get_nets {n1 n2}]
 [get_nets n3]
1
```

However if the separation is applied globally on the net for all its coupled nets, it cannot be removed by using the *-pairwise* option.

```
pt_shell> set_coupling_separation [get_nets {n4}]
1
pt_shell> remove_coupling_separation [get_nets {n4}] -pairwise [get_nets
 n3]
Warning: Cannot remove an effect that was not set on net(s) n4 n3.
 (XTALK-107)
1
```

The coupling separation can be applied on the net pairwise with all its coupled nets in the following way so that pairwise exclusions can be applied on the net with any of its coupled nets.

```
pt_shell> set_coupling_separation [get_nets VIC_NET] -pairwise
[get_attribute -class net [get_nets VIC_NET] aggressors]
1

pt_shell> remove_coupling_separation [get_nets VIC_NET] -pairwise
[get_nets AGG_NET*]
1

pt_shell> remove_coupling_separation -all
1
```

See Also

- report_si_delay_analysis
- report_si_noise_analysis
- set_coupling_separation
- write_changes

Feedback

remove_current_session

Removes the previously set session.

Syntax

status *remove_current_session*

Arguments

None.

Description

This command is available only if you invoke the pt_shell with the *-multi_scenario* option.

In the multi-scenario analysis, a list of scenarios is brought into session and command focus (selected for analysis) by using the *current_session* command. These scenarios are then defocused using the *remove_current_session* command. However, the scenarios are not removed from memory and could be set in focus for another subsequent session.

Examples

In the following example, two scenarios are in the current session. These scenarios are then removed from focus.

```
pt_shell> current_session {scen2 scen3}
1
pt_shell> remove_current_session
1
```

See Also

- report_multi_scenario_design
- remove_scenario

remove_data_check

Removes specified data-to-data checks previously set by the *set_data_check* command.

Syntax

string *remove_data_check*

```
[-from from_object
   | -rise_from from_object
   | -fall_from from_object]
[-to to_object
   | -rise_to to_object
   | -fall_to to_object]
```

```
[-setup | -hold]
[-clock clock_object]
[-sms_scenarios sms_scenarios_list]
```

Data Types

```
from_object          collection
to_object            collection
clock_object         collection
sms_scenarios_list   collection
```

Arguments

`-from from_object`

Specifies a pin or port in the current design as the related pin of the data-to-data check is removed. Both rising and falling checks are removed. You must specify one of the *-from*, *-rise_from*, or *-fall_from* options.

`-rise_from from_object`

Similar to the *-from* option, but applies only to rising delays at the related pin. You must specify one of the *-from*, *-rise_from*, or *-fall_from* options.

`-fall_from from_object`

Similar to the *-from* option, but applies only to falling delays at the related pin. You must specify one of the *-from*, *-rise_from*, or *-fall_from* options.

`-to to_object`

Specifies a pin or port in the current design as the constrained pin of the data-to-data check is created. Both rising and falling checks are removed. You must specify one of the *-to*, *-rise_to*, or *-fall_to* options.

`-rise_to to_object`

Similar to the *-to* option, but applies only to rising delays at the constrained pin. You must specify one of the *-to*, *-rise_to*, or *-fall_to* options.

`-fall_to to_object`

Similar to the *-to* option, but applies only to falling delays at the constrained pin. You must specify one of the *-to*, *-rise_to*, or *-fall_to* options.

`-setup`

Indicates that only the setup data check is removed. If neither the *-setup* or *-hold* option is specified, both setup and hold checks are removed.

`-hold`

Indicates that only the hold data check is removed. If neither the *-setup* or *-hold* is specified, both setup and hold checks are removed.

`-clock` *clock_object*

Indicates that the data check for the specified clock at the related pin is removed. This option applies only if a previous *set_data_check* command used the *-clock* option to specify the same clock; otherwise, this option is ignored.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only data checks compatible with the specified SMS scenarios. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *remove_data_check* command specifies that data-to-data checks be removed between a "from" (related) object and a "to" (constrained) object. These checks were previously set using the *set_data_check* command.

Examples

The following example removes a data-to-data check from and1/B to and1/A with respect to the rising edge of signals at and1/B. Both setup and hold checks are removed.

```
pt_shell> remove_data_check -rise_from and1/B -to and1/A
```

The following example removes setup data-to-data check between and1/B to and1/A with respect to the rising edge of signals at and1/B, coming from startpoints triggered with clock CK1, falling edge of signals at and1/A.

```
pt_shell> remove_data_check -rise_from and1/B -fall_to and1/A -setup
 -clock [get_clock CK1]
```

See Also

- report_timing
- set_data_check
- update_timing

---

remove_design

Removes one or more designs from memory.

Syntax

string *remove_design*

```
-all
-hierarchy
designs
```

Data Types

designs	list

Arguments

`-all`

Indicates that all designs are to be removed. The *-all*, *-hierarchy*, and *designs* options are mutually exclusive; you can specify only one.

`-hierarchy`

Indicates that all designs in the hierarchy of the current design are to be removed, including the current design. The *-all*, *-hierarchy*, and *designs* options are mutually exclusive; you can specify only one.

designs

Specifies a list of designs to remove. The *-all*, *-hierarchy*, and *designs* options are mutually exclusive; you can specify only one.

Description

The *remove_design* command removes designs from pt_shell. You must specify either the *-all*, *-hierarchy* or *designs* option, but not more than one of them. The *remove_design* command does not remove libraries; use the *remove_lib* command for that purpose.

Note that you cannot combine the *-hierarchy* option with a list of designs. The *-hierarchy* option is restricted to the current design. If the current design is not linked, it is automatically linked. Then, all designs in the hierarchy of the current design are removed, including the current design itself.

Examples

The following example removes the 'd1' and 'd2' designs:

```
pt_shell> remove_design {d1 d2}
Removing design 'd1'...
Removing design 'd2'...
1
```

The following example removes all designs in memory:

```
pt_shell> remove_design -all
Removing design 'd3'...
1
```

The following example removes all designs in the hierarchy of the current design. In this example, the design was already linked.

```
pt_shell> remove_design -hierarchy
Removing design 'TOP'...
Removing design 'eBlock'...
Removing design 'iBlock'...
Removing design 'mBlock'...
1
```

See Also

- current_design
- list_designs
- remove_lib

---

remove_design_mode

Removes specified design modes and/or cell mode and path mappings to design modes.

Syntax

status *remove_design_mode*

```
[-from from_pin_list]
[-to to_pin_list]
[-through through_pin_list]
mode_list
[cell_mode_list]
[instance_list]
```

Data Types

from_pin_list	list
to_pin_list	list
through_pin_list	list
mode_list	list
cell_mode_list	list
instance_list	list

Arguments

`-from` *from_pin_list*

Specifies a timing path, from a given pin of the design, that is active only for the specified design mode. The given paths are inactive for other design modes. These paths are unmapped from the specified design mode and any previous settings on the path due to the design mode are removed.

`-to` *`to_pin_list`*

Specifies a timing path to a given pin of the design, that is active only for the specified design mode. The given paths are inactive for other design modes. These paths are unmapped from the specified design mode and any previous settings on the path due to the design mode are removed.

`-through` *`through_pin_list`*

Lists the pins, ports, or nets through which the paths pass. The paths become active for the specified mode. You can specify the *-through* option more than one time in one command invocation. Nets are interpreted to imply the leaf-level driver pins. These paths are unmapped from the *design_mode and* command and any previous settings on the path due to the design mode are reset.

`mode_list`

Specifies a list of design modes to be removed. Design modes on the list must have been previously created using the *define_design_mode_group* command.

`cell_mode_list`

Specifies a list of cell modes to be unmapped from the specified design mode. Any previous settings on any cell mode due to the design mode are reset.

`instance_list`

Specifies a list of cells in which the specified modes are unmapped from the design mode. This list must be accompanied by the *cell_mode_list* option.

Description

The *remove_design_mode* command removes specified design modes previously created by the *define_design_mode_group* command or removes mappings for specified design modes created by the *map_design_mode* command.

A design mode can be removed from a design mode group using the *remove_design_mode* command. When a design mode is removed from a design mode group, all previous settings on cell modes and paths are reset. Also, if this design mode was the active design mode, then all other design modes in the design mode group are reset to default.

A path can be unmapped from a design mode using the *remove_design_mode -from [from_pin_list] -to [to_pin_list] -through [through_pin_list]* command. The pin specification must exactly match the pin specification set using a previous *map_design_mode* command. When a path is unmapped from a design mode, all paths are reset if they have been previously set false by that design mode. When a cell mode is unmapped from a design mode, all cell modes are reset if they have been previously set by that design mode.

To see a report of the design modes specified for the current design, use the *report_mode* command. The report also shows the cell modes and paths mapped to each design mode.

Examples

The following example defines two design modes, DM1 and DM2, maps the cell mode READ to design mode DM1 for instance Uram1, then removes the design mode DM1. Removing DM1 also removes the mapping to the active cell mode READ.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode READ Uram1 DM1
pt_shell> remove_design_mode DM1
```

See Also

- define_design_mode_group
- map_design_mode
- report_mode
- reset_mode
- set_mode

---

remove_disable_clock_gating_check

Restores clock gating checks, previously disabled by the *set_disable_clock_gating_check* command, for specified cells and pins.

Syntax

string *remove_disable_clock_gating_check*

```
        object_list
```

Data Types

object_list list

Arguments

object_list

Specifies a list of cells, pins, lib-pins or lib-cells for which previously-disabled clock gating checks are to be restored.

Feedback

Description

Restores timing clock gating checks previously disabled with the *set_disable_clock_gating_check* command.

Examples

The following example restores disabled clock gating checks for the object an1.

```
pt_shell> remove_disable_clock_gating_check an1/A
```

The following example restores all disabled gating checks on the cell U1/or2.

```
pt_shell> remove_disable_clock_gating_check U1/or2
```

The following example restores all disabled gating checks on the lib-cell class/AND1.

```
pt_shell> remove_disable_clock_gating_check [get_lib_cells
 {"class/AND1"}]
```

The following example restores all disabled gating checks on the lib-pin class/AND1/A.

```
pt_shell> remove_disable_clock_gating_check [get_lib_pins
 {"class/AND1/A"}]
```

See Also

- set_disable_clock_gating_check

---

remove_disable_timing

Enables the previously disabled timing arcs.

Syntax

status *remove_disable_timing*

```
[-from from_pin_name]
[-to to_pin_name]
object_list
```

Data Types

from_pin_name	string
to_pin_name	string
object_list	list

Feedback

Arguments

`-from` *from_pin_name*

Specifies that only the arcs from this pin on the specified cell or library cell are to be disabled. When used, the *-from* and *-to* options must be specified together.

`-to` *to_pin_name*

Specifies that only the arcs to this pin on the specified cell or library cell are to be disabled. When used, the *-from* and *-to* options must be specified together.

object_list

Specifies a list of cells, pins, ports, library cells, library cell pins, cell timing arcs, or library timing arcs. If the *-from* and *-to* options are specified, the *object_list* argument must contain only cells or library cells.

Description

Restore timing arcs previously disabled with the *set_disable_timing* command. When the *-from* and *-to* option pins are specified, all arcs between these two pins on the cell or library cell are restored.

To list the timing arcs for a library cell, use the *report_lib -timing_arcs* command.

Examples

The following example restores disabled setup and hold arcs between pins CLK and TE on library cell SFF1.

```
pt_shell> remove_disable_timing -from CLK -to TE tech_lib/SFF1
```

The following example restores all disabled cell arcs from or to pin A on cell U1/U2.

```
pt_shell> remove_disable_timing U1/U2/A
```

See Also

- report_disable_timing
- report_lib
- set_disable_timing

---

remove_distributed_design

Removes design netlist and associated data from the DMSA manager shell.

Syntax

status *remove_distributed_design*

Feedback

Description

The *remove_distributed_design* command will remove the design data previously loaded by the *load_distributed_design* command.

See Also

- cache_distributed_attribute_data

---

remove_dont_override

Removes user-specified *set_dont_override* commands from specified objects.

Syntax

string *remove_dont_override*

```
        object_list
```

Data Types

object_list list

Arguments

object_list

Removes user-specified *set_dont_override* commands from the specified objects.

Description

This command removes user-specified *set_dont_override* commands at the HyperScale block-level flow.

Examples

Assuming a design with two levels of hierarchy, where TOP instantiates a subblock BOT, and the named scenario of interest is fast_func. To perform hierarchical timing analysis for BOT and automatically generate a reduced and accurate timing representation for its parent level analysis, use the following commands:

```
pt_shell> set hier_enable_analysis true
pt_shell> set_hier_config -parent -name "fast_func" -path /design/TOP
pt_shell> set_hier_config -name "fast_func" -path /design/blocks/BOT
```

Feedback

In addition, for the block BOT, it is necessary to freeze the timing budget defined for the bused port DATAIN so that PrimeTime does not automatically use the actual context pushed from TOP to override your settings, use the following command:

```
pt_shell> set_dont_override [get_port DATAIN*]
```

The following example removes the previous user-specified set_dont_override:

```
pt_shell> remove_dont_override [get_port DATAIN*]
```

See Also

- link_design
- report_constraint
- save_session
- set_dont_override
- set_hier_config
- update_timing
- hier_enable_analysis

---

remove_drc_error_data

Removes DRC error data objects.

Syntax

```
status remove_drc_error_data
drc_error_data
[-force]
```

Data Types

```
drc_error_data   collection
```

Arguments

drc_error_data

Specifies the collection of DRC error data objects to remove.

-force

Removes the specified DRC error data object, regardless of pending multiple opens that have not been matched with a close.

Feedback

Description

The *remove_drc_error_data* command removes the specified DRC error data objects. The command first closes the error data objects, which decrements the open count for the error data objects. If the open count decrements to zero, the command removes the error data object from memory. If the error data object was created as an external error data file, the error data file is removed from disk.

The open count for a DRC error data object is incremented each time you run the *create_drc_error_data* or *open_drc_error_data* command on it. The open count is decremented each time you run the *close_drc_error_data* command. The error data object is removed from memory when the open count reaches zero.

Use the *-force* option to remove the error data object from memory and remove the external error data file from disk, regardless of the open count value.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example removes the DRC error data object named "dppinassgn.err".

```
prompt> remove_drc_error_data dppinassgn.err
1
```

The following example first opens an error data object named "dppinassgn.err", and then deletes it.

```
prompt> set data [open_drc_error_data -all dppinassgn.err]
{"dppinassgn.err"}
prompt> remove_drc_error_data $data
1
```

See Also

- close_drc_error_data
- create_drc_error_data
- get_drc_error_data
- open_drc_error_data
- save_drc_error_data

---

remove_drc_error_types

Removes physical DRC error type objects.

Feedback

Syntax

```
status remove_drc_error_types
-error_data drc_error_data
drc_error_types
```

Data Types

```
drc_error_data     collection
drc_error_types    collection
```

Arguments

`-error_data drc_error_data`

Specifies the error data from which to remove physical DRC error types.

drc_error_types

A collection of physical DRC error type objects to remove.

Description

The *remove_drc_error_types* command removes physical DRC error type objects from the given error data. When an error type is removed, all errors of the type are also deleted from the error data.

Examples

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes all error_types that have the error_class value of *short*.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> remove_drc_error_types -error_data $data \\
   [get_drc_error_types -error_data $data -filter {error_class==short}]
```

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes the error type named "pin short".

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> remove_drc_error_types -error_data $data \\
   [get_drc_error_types -error_data $data {"pin short"}]
```

See Also

- create_drc_error_type
- get_drc_error_types

Feedback

remove_drc_errors

Removes phyiscal DRC error data objects

Syntax

```
status remove_drc_errors
-error_data drc_error_data
drc_errors
```

Data Types

```
drc_error_data      collection
drc_errors          collection
```

Arguments

`-error_data drc_error_data`

Specifies the error data from which to remove errors.

drc_errors

A collection of physical DRC error data objects to remove.

Description

The *remove_drc_errors* command removes physical DRC error objects from the given error data.

Examples

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes all errors that have the status value of *fixed*.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
prompt> remove_drc_errors -error_data $data \\
        [get_drc_errors -error_data $data -filter {status==fixed}]
```

The following example opens a physical DRC error data file that is named "my_design_dppinassgn.err", then removes all errors that are of type "pin short".

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{"my_design_dppinassgn.err"}
```

```
prompt> set pinShortType [get_drc_error_types -error_data $data {"pin
 short"}]
{"pin short"}
prompt> remove_drc_errors -error_data $data \\
        [get_drc_errors -error_data $data -of_objects $pinShortType]
```

See Also

- create_drc_error
- remove_drc_error_types
- remove_drc_error_data
- open_drc_error_data

---

remove_drive_resistance

Removes drive resistance for input or inout ports.

Syntax

string *remove_drive_resistance*

```
        port_list
```

Data Types

port_list list

Arguments

port_list

Specifies a list of input or inout port names of the current design from which the drive values are to be removed.

Description

The *remove_drive_resistance* command removes the drive resistance value from one or more input or inout ports. This is equivalent to using the *set_drive_resistance 0* command. In fact, that is how this command works. So, if output ports are passed into the *remove_drive_resistance* command, a DES-003 message is issued, indicating that the *set_drive_resistance* command could not be performed on those ports.

Feedback

Examples

The following command shows the difference in the output report after using the *remove_drive_resistance* command:

```
pt_shell> set_drive_resistance 5 [get_ports {A B C}]
1
pt_shell> report_port -drive {A B C}

****************************************
Report : port
         -drive
Design : counter
****************************************


                  Resistance          Transition
Input Port     Rise      Fall      Rise      Fall
------------------------------------------------------------
A              5.00      5.00      --        --
B              5.00      5.00      --        --
C              5.00      5.00      --        --

1
pt_shell> remove_drive_resistance A
1
pt_shell> report_port -drive {A B C}

****************************************
Report : port
         -drive
Design : counter
****************************************


                  Resistance          Transition
Input Port     Rise      Fall      Rise      Fall
------------------------------------------------------------
A               --        --       --        --
B              5.00      5.00      --        --
C              5.00      5.00      --        --

1
```

See Also

- report_port
- set_load
- set_drive_resistance

Feedback

remove_driving_cell

Removes port driving cell information.

Syntax

string *remove_driving_cell*

```
[-rise]
[-fall]
[-min]
[-max]
[-clock clock_name]
[-clock_fall]
port_list
```

Data Types

clock_name	string
port_list	list

Arguments

`-rise`

Removes rise driving cell information.

`-fall`

Removes fall driving cell information.

`-min`

Removes min driving cell information.

`-max`

Removes max driving cell information.

`-clock clock_name`

Removes the driving cell set relative to the specified clock.

`-clock_fall`

Removes the driving cell relative to the falling edge of the clock. The default is the rising edge.

port_list

Provides a list of input or output ports.

Feedback

Description

Removes driving cell information from the specified ports. The driving cell information is specified with the *set_driving_cell* command. The default is to remove all the *-rise* and *-fall* option information.

To see port drive information, use the *report_port -drive* command.

Examples

The following example removes all driving cell information for port IN2.

```
pt_shell> remove_driving_cell IN2
```

See Also

- report_port
- set_driving_cell

---

remove_dvd

Removes dynamic voltage drop information from the current design.

Syntax

status *remove_dvd*

```
[-rail_map_only]
```

Arguments

`-rail_map_only`

Removes rail map scaling/offset values applied by *read_dvd -rail_map*, while still leaving the voltage drop values themselves in place.

This option can be used to efficiently move the analysis from one scaled corner to another.

Description

Removes user-specified dynamic voltage drop information that was previously applied with the *read_dvd* command.

Examples

The following example removes all dynamic voltage drop information from the design:

```
pt_shell> remove_dvd
```

Feedback

The following example removes only the rail-map scaling/offset information:

```
pt_shell> remove_dvd -rail_map_only
```

See Also

- read_dvd
- report_timing
- report_voltage_robustness
- timing_enable_dvd_analysis

---

remove_fanout_load

Removes fanout load information from output ports in the current design.

Syntax

status *remove_fanout_load*

```
[-sms_scenarios sms_scenarios_list]
port_list
```

Data Types

```
port_list            list
sms_scenarios_list   collection
```

Arguments

port_list

Specifies a list of output ports.

-sms_scenarios *sms_scenarios_list*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the fanout load compatible with the specified SMS scenarios. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

Description

Removes fanout load information specified by the *set_fanout_load* command. Fanout load for a net is the sum of the *fanout_load* attributes for all input pins and output ports connected to the net. Output pins can have maximum fanout limits, specified in the library or with the *set_max_fanout* command. The *fanout_load* is used when checking max_fanout design values.

Feedback

Use the *report_constraint* command with the *-max_fanout* option to show maximum fanout constraint evaluations. Use the *report_port* command with the *-design_rule* option to show port fanout load values.

Examples

The following example removes the fanout load on ports matching "OUT*".

```
pt_shell> remove_fanout_load OUT*
```

See Also

- report_constraint
- report_port
- set_fanout_load
- set_max_fanout

---

remove_from_collection

Removes objects from a collection, resulting in a new collection. The base collection remains unchanged.

Syntax

collection *remove_from_collection*

```
[-intersect]
collection1
object_spec
```

Data Types

collection1	collection
object_spec	list

Arguments

`-intersect`

Removes objects from collection1 not found in object_spec. Without this option, removes objects from collection1 that are found in object_spec.

collection1

Specifies the base collection to be copied to the result collection. Objects matching *object_spec* are removed from the result collection.

object_spec

Specifies a list of named objects or collections to remove. The object class of each element in this list must be the same as in the base collection. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

Description

The *remove_from_collection* command removes elements from a collection, creating a new collection.

If the base collection is homogeneous, any element of the *object_spec* that is not a collection is searched for in the database using the object class of the base collection. If the base collection is heterogeneous, any element of the *object_spec* that is not a collection is ignored.

If the *-intersect* option is not specified, which is the default mode, and if nothing matches the *object_spec*, the resulting collection is a copy of the base collection. If everything in the *collection1* option matches the *object_spec*, the result is the empty collection. With the *-intersect* option the results are reversed.

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

Examples

The following example from PrimeTime gets all input ports except "CLOCK".

```
pt_shell> set cPorts [remove_from_collection [all_inputs] CLOCK]
{"in1", "in2"}
```

See Also

- add_to_collection
- collections

---

remove_generated_clock

Removes generated clock objects from the current design.

Syntax

status *remove_generated_clock*

```
-all
clock_list
```

Data Types

```
clock_list        list
```

Feedback

Arguments

`-all`

Indicates that all generated clocks are to be removed.

`clock_list`

Specifies a list of names of generated clocks to be removed.

Description

This command removes from the current design generated clocks previously created using the *create_generated_clock* command. All attributes set on generated clocks such as, *false_paths* and *multicycle_paths*, are also removed.

To display information about clocks and generated clocks in the design, use the *report_clock* command.

Examples

The following example removes the generated clock GEN1 from the design.

```
pt_shell> remove_generated_clock GEN1
```

The following example removes all generated clocks from the design.

```
pt_shell> remove_generated_clock -all
```

See Also

- create_generated_clock
- report_clock

---

remove_host_options

Removes hosts options set using the *set_host_options* command.

Syntax

int *remove_host_options*

`[`*`host_option_names`*`]`

Data Types

`host_option_names`	`list`

Feedback

Arguments

host_option_names

This option specifies a list of named host options to remove.

Description

Given a list of host option names, removes those host options. The host option names specifed must have been previously set using the *set_host_options* command. If no host option names are specified, all host options are removed. All processes belonging to the host options being removed are shutdown.

In a DMSA manager process, if a host option is removed then it will also be removed from the affinity specification of all scenarios. If all host options are removed then all scenario affinities will be removed. Where scenarios specify an affinity the removal of host opitons will reduce the set of workers upon which the scenario can run.

Examples

In the following example, the manager process is running on the host named ptnlm15 using a 64-bit binary and specifies several different sets of host options.

The first host options, named "my_opts1", specifies

```
- to use one process on the same host as the manager process
- no upper limit on the number of cores to use so the default applies.
```

The second host options, named "my_opts2", specifies

```
- to use one process on the same host as the manager, explicitly
  mentioning the manager host by name
- to use a maximum of two cores.
```

The third host options, named "my_opts3", specifies

```
- to use one process on the host named ptnlm11, since no submit
  command is specified, "rsh" will be used to connect to ptnlm11
- to use a maximum of three cores.
```

The fourth host options named "my_opts4", specifies

```
- to use one process on the host named ptnlm12 connecting to
  ptnlm12 using the submit command "/usr/bin/rsh"
- no upper limit on the numebr of cores to use so the default applies.
```

The fifth host options named "my_opts5", specifies

```
- to use one process on an LSF farm, using the "bsub" command found
  from the environment path to submit the job, requiring 1GB of
  memory and two slots per compute server. Notice that the farm job
  must be specified with the number of slots needed to support the
  number of cores to use with the -n 2 -R span\\[ptile=2\\]
```

Feedback

```
  options.
- to use the terminate command "bkill" found from the environment
  path for the termination of jobs that do not come online.
- to use a maximum of two cores.
```

The sixth host options named "my_opts6", specifies

```
- to use one process on a Grid farm, using the "qsub" command found
  from the environment path to submit the job, using the project
  named "bnormal", requiring 1GB of memory and four slots per compute
  server. Notice that the farm job must be specified with the number
  of slots needed to support the number of cores to use by targeting
  the parallel environment "mt" requesting 4 slots with the "-pe mt 4"
  options.
- to use the terminate command "qdel" found from the environment
  path for the termination of jobs that do not come online.
- to use a maximum of four cores.
```

Note: After the *remove_host_options* command has been called all processes are shutdown and all the host options are removed.

Call the *set_host_options* command to specify the host options.

```
pt_shell> set_host_options -name my_opts1 -num_processes 1
1
pt_shell> set_host_options -name my_opts2 -num_processes 1 \\
          ptnlm15 -max_cores 2
1
pt_shell> set_host_options -name my_opts3 -num_processes 1 \\
          ptnlm11 -max_cores 3
Warning: no submit command specified to access remote host
 'ptnlm10'. The 'rsh' command will be used. (CMCR-004)
1
pt_shell> set_host_options -name my_opts4 -num_processes 1 \\
          -submit_command "/usr/bin/rsh" ptnlm12
1
pt_shell> set_host_options -name my_opts5 -num_processes 1 \\
          -submit_command "[sh which bsub] -n 2 -R rusage\[mem=1000\] \\
          -R span\[ptile=2\]" -terminate_command "[sh which bkill]" \\
          -max_cores 2
1
pt_shell> set_host_options -name my_opts6 -num_processes 1 \\
          -submit_command "[sh which qsub] -P bnormal -pe mt 4 \\
          -l mem_free=1G" -terminate_command "[sh which qdel]" \\
          -max_cores 4
1
```

Call the *start_hosts* command to start the worker processes. Notice that each instance launched is assigned a number e.g. 1] Launched ...

```
pt_shell> start_hosts
Launching 6 Distributed Workers
```

```
   1] Launched : /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 4

   2] Launched : ptnlm15 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 2

   3] Launched : rsh ptnlm11 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 3

   4] Launched : /usr/bin/rsh ptnlm12 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 4

   5] Launched : /lsf/bin/bsub -n 2 -R rusage[mem=1000] -R span[ptile=2]
 \\
                 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 2

   6] Launched : /remote/sge2/default/bin/lx-amd64/qsub -P bnormal \\
                 -pe mt 4 -l mem_free=1G /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 4


 ------------------------------------------------------------------------
---
  Distributed farm creation timeout : 21600 seconds
  Waiting for   6 (of   6) distributed hosts (Tue Nov 25 04:39:34 2014)
  Waiting for   2 (of   6) distributed hosts (Tue Nov 25 04:39:49 2014)
  Waiting for   0 (of   6) distributed hosts (Tue Nov 25 04:39:59 2014)

 ------------------------------------------------------------------------
---

1
```

Call the *report_host_usage* command after starting the hosts to report the host options specifed by the *set_host_options* command, as well as the real time data associated with the processes.

```
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 04:59:46 2014
****************************************

  Options Name          Host Name              Num Processes
  ------------------------------------------------------------------
  my_opts1              **localhost**          1
  my_opts2              **ptnlm15**            1
  my_opts3              ptnlm11                1
  my_opts4              ptnlm12                1
```

```
my_opts5              >>farm<<              1
my_opts6              >>farm<<              1

Options Name      #    Host Name    Job ID    Process ID    Status
-------------------------------------------------------------------
my_opts1          1                 27337     27337         ONLINE
my_opts2          2    ptnlm15      27378     27378         ONLINE
my_opts3          3    ptnlm11      23716     23716         ONLINE
my_opts4          4    ptnlm12      13059     13059         ONLINE
my_opts5          5    maddog136    312487    21901         ONLINE
my_opts6          6    peemt32      2165476   5293          ONLINE

Usage limits (cores)

Options Name      #                                          Effective
-------------------------------------------------------------------
(local process)   -                                          4
my_opts1          1                                          -
my_opts2          2                                          -
my_opts3          3                                          -
my_opts4          4                                          -
my_opts5          5                                          -
my_opts6          6                                          -
-------------------------------------------------------------------
Total                                                        4

Memory usage

Options Name      #                             Memory (MB)
-------------------------------------------------------------------
(local process)   -                             445.43
my_opts1          1                             0.00
my_opts2          2                             0.00
my_opts3          3                             0.00
my_opts4          4                             0.00
my_opts5          5                             0.00
my_opts6          6                             0.00

Performance

Options Name      #    CPU Time (s)             Elapsed Time (s)
-------------------------------------------------------------------
(local process)   -    2                        1023
my_opts1          1    0                        0
my_opts2          2    0                        0
my_opts3          3    0                        0
my_opts4          4    0                        0
my_opts5          5    0                        0
my_opts6          6    0                        0
```

1

Feedback

Call the *remove_host_options* command to remove all host options. Note: After the *remove_host_options* command has been called all processes are shutdown and all the host options are removed.

```
pt_shell> remove_host_options
Shutting down worker processes ...
 Shutdown Process 1
 Shutdown Process 2
 Shutdown Process 3
 Shutdown Process 4
 Shutdown Process 5
 Shutdown Process 6
1
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 06:58:46 2014
****************************************


  Usage limits (cores)

  Options Name     #                                                Effective
  -----------------------------------------------------------------------
  (local process)  -                                                4
  -----------------------------------------------------------------------
  Total                                                             4

  Memory usage

  Options Name     #                            Memory (MB)
  -----------------------------------------------------------------------
  (local process)  -                            330.80

  Performance

  Options Name     #    CPU Time (s)            Elapsed Time (s)
  -----------------------------------------------------------------------
  (local process)  -    2                       52

1
```

See Also

- set_host_options
- report_host_usage
- start_hosts
- stop_hosts

Feedback

remove_ideal_aggressor

Removes ideal aggressor marking on the specified ideal aggressor nets. Removes the *si_ideal_aggressor* attribute on these nets.

Syntax

int *remove_ideal_aggressor*

```
-all
net_list
```

Data Types

```
net_list    list
```

Arguments

`-all`

Indicates that ideal aggressor marking on all nets of the current design is to be removed. The *-all* and *net_list* options are mutually exclusive; you can specify only one.

net_list

Specifies a list of ideal aggressor nets.

Description

This command removes ideal aggressor merking on the specified nets of the current design. The *si_ideal_aggressor* attribute on these nets is removed.

See the *set_ideal_aggressor* command for more details on ideal aggressor handling.

You can query the *si_ideal_aggressor* attribute by using the *get_attribute* command to confirm that the attribute is successfully removed.

Examples

The following command removes ideal aggressor marking on ideal aggressor nets "n_ideal_aggr1" and "n_ideal_aggr2".

```
pt_shell> remove_ideal_aggressor [get_nets {n_ideal_aggr1 n_ideal_aggr2}]
1
```

In the following example, the *si_ideal_aggressor* attribute is set using *set_ideal_aggressor* command and then queried using *get_attribute* command. The *get_attribute* command returns *true* for the *si_ideal_aggressor* attribute. The command *remove_ideal_aggressor -all* is used to remove ideal aggressor marking on all nets of the design. The *get_attribute* command then returns *false* for the *si_ideal_aggressor* attribute.

Feedback

```
pt_shell> get_attribute [get_net n_ideal_aggr] ref_name
n_ideal_aggr

pt_shell> get_attribute [get_net n_ideal_aggr] si_ideal_aggressor
false

pt_shell> set_ideal_aggressor [get_net n_ideal_aggr]
1

pt_shell> get_attribute [get_net n_ideal_aggr] si_ideal_aggressor
true

pt_shell> remove_ideal_aggressor -all
1

pt_shell> get_attribute [get_net n_ideal_aggr] si_ideal_aggressor
false
```

See Also

- set_ideal_aggressor
- get_attribute

---

remove_ideal_latency

Removes ideal latency values from the specified objects.

Syntax

int *remove_ideal_latency*

```
[-rise]
[-fall]
[-min]
[-max]
object_list
```

Data Types

object_list	list

Arguments

`-rise`

Specifies that only rise ideal latency should be removed. By default, both rise and fall ideal latency are removed.

Feedback

`-fall`

Specifies that only fall ideal latency should be removed. By default, both rise and fall ideal latency are removed.

`-min`

Specifies that only minimum ideal latency should be removed. By default, both minimum and maximum ideal latency are removed.

`-max`

Specifies that only maximum ideal latency should be removed. By default, both minimum and maximum ideal latency are removed.

`object_list`

Specifies a list of ports or pins from where to remove ideal latency.

Description

Removes the user-specified ideal latency that was previously set by the *set_ideal_latency* command from specified objects in the *object_list* argument.

You can remove selected portions of ideal latency by specifying applicable *-rise*, *-fall*, *-min*, and *-max* options.

To list ideal latency values, use the *report_ideal_network* command.

Examples

The following example removes ideal latency from the output pin named *Z* of cell instance *U1/U2/U3*.

```
pt_shell> remove_ideal_latency {U1/U2/U3/Z}
```

The following example removes only rise ideal latency information from a port named *A*.

```
pt_shell> remove_ideal_latency -rise {A}
```

See Also

- remove_ideal_network
- remove_ideal_transition
- report_ideal_network
- report_timing
- set_ideal_network
- set_ideal_latency

Feedback

remove_ideal_network

Removes sources of ideal networks in the current design. Cells and nets in the transitive fanout of the specified objects are no longer treated as ideal.

Syntax

int *remove_ideal_network*

```
        object_list
```

Data Types

object_list	list

Arguments

object_list

Specifies a list of ideal sources. The source objects can be ports or pins of leaf cells at any hierarchical level of the design. If nets are specified in the *object_list* option, all of the nets' global driver ideal source pins that were set with the *-no_propagate* option are removed.

Description

This command removes the ideal network property from a set of ports or pins in the design that were previously marked as ideal sources using the *set_ideal_network* command. You specify only the source of the network. The ideal network is automatically updated by PrimeTime. The criteria for propagating the ideal property are detailed in the *set_ideal_network* man page.

All ideal networks are removed using the *reset_design* command.

Examples

The following example sets up an ideal network on objects in the design CLOCK_GEN and then removes part of the network:

```
pt_shell> current_design CLOCK_GEN
pt_shell> set_ideal_network {port1 port2}
pt_shell> remove_ideal_network port2
```

The following example creates an ideal network and then removes part of the ideal network.

```
pt_shell> current_design {TOP}
pt_shell> set_ideal_network {IN1 IN2}
pt_shell> remove_ideal_network {IN1}
```

Feedback

The following example removes an ideal network that was set on a net.

```
pt_shell> set_ideal_network -no_propagate {netB}
Warning: Transferring ideal net attribute onto driver pin 'AN2/Z' of net
 'netB'. (UITE-450)
pt_shell> remove_ideal_network {netB}
```

See Also

- remove_ideal_latency
- remove_ideal_transition
- report_ideal_network
- reset_design
- set_ideal_network

---

remove_ideal_transition

Removes ideal transition values from the specified objects.

Syntax

int *remove_ideal_transition*

```
[-rise]
[-fall]
[-min]
[-max]
object_list
```

Data Types

object_list	list

Arguments

`-rise`

Specifies that only rise ideal transition should be removed. By default, both rise and fall ideal transitions are removed.

`-fall`

Specifies that only fall ideal transition should be removed. By default, both rise and fall ideal transitions are removed.

`-min`

Specifies that only minimum ideal transition should be removed. By default, both minimum and maximum ideal transitions are removed.

Feedback

`-max`

Specifies that only maximum ideal transition should be removed. By default, both minimum and maximum ideal transitions are removed.

`object_list`

Specifies a list of ports or pins from which to remove ideal transition.

Description

Removes the user-specified ideal transition that was previously set by the *set_ideal_transition* command from specified objects in the *object_list* option.

You can remove selected portions of ideal transition by specifying applicable *-rise*, *-fall*, *-min*, and *-max* options.

To list ideal transition values, use the *report_ideal_network* command.

Examples

The following example removes ideal transition from the output pin named *Z* of cell instance *U1/U2/U3*.

```
pt_shell> remove_ideal_transition {U1/U2/U3/Z}
```

The following example removes only rise ideal transition information from a port named *A*.

```
pt_shell> remove_ideal_transition -rise {A}
```

See Also

- remove_ideal_latency
- remove_ideal_network
- report_ideal_network
- report_timing
- set_ideal_network
- set_ideal_transition

---

remove_input_delay

Removes input delay information from ports or pins.

Syntax

string *remove_input_delay*

```
[-clock clock_name]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-sms_scenarios sms_scenarios_list]
port_pin_list
```

Data Types

```
clock_name           list
port_pin_list        list
sms_scenarios_list   collection
```

Arguments

`-clock clock_name`

Relative clock; '' for no clock. Use this option to remove only input delay relative to one clock.

`-clock_fall`

Delay is relative to falling edge of clock.

`-level_sensitive`

Delay is from level-sensitive latch.

`-rise`

Removes rising input delay.

`-fall`

Removes falling input delay.

`-max`

Removes maximum input delay.

`-min`

Removes minimum input delay.

`port_pin_list`

Specifies a list of ports and pins.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the input delay compatible with the specified

Feedback

SMS scenarios. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

Description

Removes input delay information from ports or pins in the current design. Input delay is specified with the *set_input_delay* command. To show input delays on ports, use the *report_port -input_delay* command. The default is to remove all input delay information in the *port_pin_list* option.

Examples

The following example removes all input delay from ports IN*.

```
pt_shell> remove_input_delay [get_ports IN*]
```

The following example removes input delay relative to CLK rise edge from port DATA[5].

```
pt_shell> remove_input_delay -clock CLK {DATA[5]}
```

See Also

- report_port
- set_input_delay
- set_output_delay
- remove_output_delay

---

remove_input_noise

Removes input noise for a library pin or port.

Syntax

int *remove_input_noise*

```
[-above]
[-below]
[-low]
[-high]
object_list
```

Data Types

list *object_list*

Feedback

Arguments

`-above`

Removes the input noise for above ground or power rail noise analysis region.

`-below`

Removes the input noise for below ground or power rail noise analysis region.

`-low`

Removes the input noise for ground rail noise.

`-high`

Removes the input noise for power rail noise. 1.0.

`object_list`

Specifies a list of lib-pins or ports.

Description

This command removes the input noise information set by the *set_input_noise* command.

Examples

This example removes input noise information for the above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_input_noise -above lsi_10k/IV/A
```

See Also

- set_input_noise

---

remove_ivm

Removes the user-defined via variation tables.

Syntax

```
status remove_ivm
   [-layer_name string]
```

Data Types

```
string     string
```

Feedback

Arguments

`-layer_name` *string*

Specifies the name of the layer. By default, the command removes via variation tables for all layers.

Description

This command removes the via variation tables previously read in with the *read_ivm* command. To remove via variation for only a specific layer, specify the layer name. To remove all via variation data, use the *read_ivm* command by itself, without arguments.

Examples

The following command removes the via variation data for only the V2 layer.

```
pt_shell> remove_ivm -layer_name V2
```

See Also

- read_ivm
- report_ivm
- timing_enable_via_variation

---

remove_lib

Removes one or more libraries from memory.

Syntax

string *remove_lib*

```
-all
libraries
```

Data Types

libraries	list

Arguments

`-all`

Removes all libraries.

libraries

Provides a list of libraries to remove.

Feedback

Description

This command removes a list of libraries. You must specify either the *libraries* or *-all* option. For a library to be removed, none of its library cells can be instantiated in any design, nor can any environment information (such as operating conditions or wire load models) be in use from the library. If this is the case, a message is issued and you must remove the design by first using the *remove_design* command and then using the *remove_lib* command.

Examples

The following command removes all the libraries read in.

```
pt_shell> remove_lib -all
Removing library '/u/libraries/cmos1.db:cmos1'...
1
```

The following command removes a library that was part of a max/min pair created by the *set_min_library* command.

```
pt_shell> remove_lib lib_ff
Removed max/min library relationship:
  Max: /u/libraries/lib_ss.db:lib_ss
  Min: /u/libraries/lib_ff.db:lib_ff
Removing library '/u/libraries/lib_ff.db:lib_ff'...
1
```

See Also

- list_libs
- list_designs
- remove_design
- set_min_library

---

remove_license

Checks in a licensed feature.

Syntax

```
status remove_license
   [-keep num_licenses]
   feature_list
```

Data Types

```
num_licenses     integer
feature_list     list
```

Feedback

Arguments

`-keep` *num_licenses*

Specifies the number of licenses to retain for each feature after the command has completed. If you do not use this option, the command checks in all of the features licenses. Incremental licensing must be enabled if you want to reduce number of licenses to a number larger than zero.

feature_list

Removes the licenses of the specified list of features. To specify more than one feature, enclose the list in curly braces ({ }). For a list of all features that are available at your site, see the key file.

Description

This command removes (checks in) the specified licensed features from the features you are currently using.

For multicore runs, where multiple licenses might be required for certain features, you can use the *-keep* option to restrict the number of licenses that are checked in. This is analogous to the *get_license -quantity* command.

Reducing the number of licenses to a number larger than zero requires incremental licensing to be enabled. Incremental licensing is enabled by default. To disable it, set the *SNPSLMD_ENABLE_INCREMENTAL_LICENSE_HANDLING* UNIX enviroment variable to *FALSE*.

To list the features that you currently have checked out, run the *list_licenses* command.

Examples

This example checks in the PrimePower and PrimeTime SI licenses.

```
pt_shell> remove_license {PrimePower PrimeTime-SI}
Checked in license 'PrimePower' (PT-020)
Checked in license 'PrimeTime-SI' (PT-020)
1
```

See Also

- get_license
- license_users
- list_licenses

---

remove_license_limit

Removes the limit on the number of licenses of given features that can be checked out.

Feedback

Syntax

```
status remove_license_limit
  feature_list
```

Data Types

feature_list	list

Arguments

feature_list

Specifies a list of features for which limits should be removed.

Description

The *remove_license_limit* command removes the upper limit on the number of licenses of given features that can be checked out previously set by the *set_license_limit* command.

By default, the command removes all limits set for any feature. If the optional *feature_list* is given, it removes the limits only for the features given in the list.

Examples

In the following example, the license limit for PrimeTime feature is removed.

```
pt_shell> remove_license_limit PrimeTime
Information: Removing license limit for feature PrimeTime. (PT-024)
1
```

See Also

- get_license
- list_licenses
- remove_license
- report_license_limit
- report_multi_scenario_design
- set_license_limit

---

remove_max_area

Removes the *max_area* attribute from the current design.

Syntax

int *remove_max_area*

Feedback

Arguments

None.

Description

The *remove_max_area* command removes the *max_area* attribute from the current design. This attribute represents the target area of the design.

Use the *set_max_area* command to set the *max_area* attribute on the current design.

Use the *report_constraint* command to show area cost of the design.

Examples

The following example removes the target area from the current design.

```
pt_shell> remove_max_area
1
pt_shell> get_attribute [get_design [current_design]] max_area
Warning: Attribute 'max_area' does not exist on design 'TOP' (ATTR-3)
```

See Also

- current_design
- get_attribute
- set_max_area
- report_constraint
- reset_design

---

remove_max_capacitance

Removes maximum capacitance limits from pins, library pins, ports, clocks, or designs.

Syntax

string *remove_max_capacitance*

```
[-sms_scenarios sms_scenarios_list]
object_list
```

Data Types

object_list	list
sms_scenarios_list	collection

Feedback

Arguments

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the maximum capacitance limits compatible with the specified SMS scenarios. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

`object_list`

Provides a list of pins, library pins, ports, clocks, or designs from which to remove maximum capacitance limits.

Description

This command removes maximum capacitance limits from pins, library pins, ports, clocks, or designs. A maximum capacitance limit is specified with the *set_max_capacitance* command.

The *report_constraint -max_capacitance* command shows maximum capacitance constraint evaluations. The *report_port -design_rule* command shows port maximum capacitance limits. The *report_design* command shows the default maximum capacitance limit for the current design.

Examples

The following example removes the maximum capacitance limit from ports "OUT*".

```
pt_shell> remove_max_capacitance [get_ports "OUT*"]
```

The following example removes the maximum capacitance limit from the current design.

```
pt_shell> remove_max_capacitance [current_design]
```

See Also

- current_design
- get_ports
- remove_min_capacitance
- report_constraint
- report_design
- report_port
- set_max_capacitance

Feedback

remove_max_fanout

Removes maximum fanout limits from ports or designs.

Syntax

string *remove_max_fanout*

```
[-sms_scenarios sms_scenarios_list]
object_list
```

Data Types

```
object_list          list
sms_scenarios_list   collection
```

Arguments

object_list

Lists the ports or designs from which to remove maximum fanout limits.

-sms_scenarios *sms_scenarios_list*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the maximum fanout limits compatible with the specified SMS scenarios. This option is only available with SMVA/SMC analysis. This collection is created by *get_sms_scenarios*.

Description

Removes maximum fanout limits from ports or designs. A maximum fanout limit is specified with the *set_max_fanout* command.

The *report_constraint -max_fanout* command shows maximum fanout constraint evaluations. The *report_port -design_rule* command shows port maximum fanout limits. The *report_design* command shows the default maximum fanout setting for the current design.

Examples

The following example removes the maximum fanout limit on ports "OUT*".

```
pt_shell> remove_max_fanout [get_ports "OUT*"]
```

The following example removes the maximum fanout limit on the current design.

```
pt_shell> remove_max_fanout [current_design]
```

Feedback

See Also

- current_design
- get_ports
- report_constraint
- report_design
- report_port
- set_fanout_load
- set_max_capacitance
- set_max_fanout
- set_max_transition

---

remove_max_time_borrow

Removes time borrow limit for latches.

Syntax

string *remove_max_time_borrow*

```
        object_list
```

Data Types

```
object_list        list
```

Arguments

object_list

Lists clocks, cells, data pins, or clock (enable) pins. If you specify a cell, all enable pins on that cell are affected.

Description

Removes maximum time borrow limit on objects. Time borrowing limits are specified with the *set_max_time_borrow* command. When the limit is removed, full time borrowing is allowed on level-sensitive latches.

To show time borrow limits, use the *report_exceptions* command.

Feedback

Examples

The following example removes the maximum time borrow limit on clock PHI1.

```
pt_shell> remove_max_time_borrow [get_clocks PHI1]
```

The following example removes the maximum time borrow limit on all the pins of cells matching U1/latch*.

```
pt_shell> remove_max_time_borrow [get_cells U1/latch*]
```

See Also

- report_exceptions
- set_max_time_borrow

---

remove_max_transition

Removes maximum transition limits from pins, ports, clocks, or designs.

Syntax

string *remove_max_transition*

```
[-sms_scenarios sms_scenarios_list]
object_list
```

Data Types

object_list	list
sms_scenarios_list	collection

Arguments

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the maximum transition limits compatible with the specified SMS scenarios. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

object_list

Provides a list of pins, ports, clocks, or designs from which to remove maximum transition limits.

Description

This command removes maximum transition limits from pins, ports, clocks or designs. A maximum transition limit is specified with the *set_max_transition* command.

The *report_constraint -max_transition* command shows maximum transition constraint evaluations. The *report_port -design_rule* command shows port maximum transition limits. The *report_design* command shows the default maximum transition limit for the current design.

Examples

The following example removes the maximum transition limit from ports "OUT*".

```
pt_shell> remove_max_transition [get_ports "OUT*"]
```

The following example removes the maximum transition limit from the current design.

```
pt_shell> remove_max_transition [current_design]
```

See Also

- current_design
- get_ports
- report_constraint
- report_design
- report_port
- set_max_capacitance
- set_max_fanout
- set_max_transition

---

remove_min_capacitance

Removes minimum capacitance limits from ports or designs.

Syntax

string *remove_min_capacitance*

```
[-sms_scenarios sms_scenarios_list]
object_list
```

Data Types

object_list	list
sms_scenarios_list	collection

Feedback

Arguments

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the minimum capacitance limits compatible with the specified SMS scenarios. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

object_list

Provides a list of ports or designs from which to remove minimum capacitance limits.

Description

This command removes minimum capacitance limits from ports or designs. A minimum capacitance limit is specified with the *set_min_capacitance* command.

The *report_constraint -min_capacitance* command shows minimum capacitance constraint evaluations. The *report_port -design_rule* command shows port minimum capacitance limits. The *report_design* command shows the default minimum capacitance limit for the current design.

Examples

The following example removes the minimum capacitance limit from ports "OUT*".

```
pt_shell> remove_min_capacitance [get_ports "OUT*"]
```

The following example removes the minimum capacitance limit from the current design.

```
pt_shell> remove_min_capacitance [current_design]
```

See Also

- current_design
- get_ports
- remove_max_capacitance
- report_constraint
- report_design
- report_port
- set_min_capacitance
- set_max_capacitance

Feedback

- set_max_fanout
- set_max_transition

---

remove_min_pulse_width

Removes a previously-specified minimum pulse width constraint from specified design objects.

Syntax

string *remove_min_pulse_width*

```
[-low]
[-high]
[object_list]
```

Data Types

object_list list

Arguments

`-low`

Indicates that the minimum pulse width constraint is to be removed only for low clock signal levels. If you do not specify the *-low* or *-high* option, the constraint is removed for both low and high pulses of clock signals.

`-high`

Indicates that the minimum pulse width constraint is to be removed only for high clock signal levels. If you do not specify the *-low* or *-high* option, the constraint is removed for both low and high pulses of clock signals.

object_list

Specifies a list of clocks, cells, pins, or ports in the current design for which the minimum pulse width constraint is to be removed. If you specify a cell, all pins on that cell are affected. If you do not specify any objects, the minimum pulse width check is removed from all objects in the current design.

Description

The *remove_min_pulse_width* command removes the pulse width check previously specified by the *set_min_pulse_width* command for clock signals in clock trees or at sequential devices.

To generate a report of pulse width constraints, use the *report_constraint -min_pulse_width* or *report_min_pulse_width* command.

The *reset_design* command removes all user-specified attributes from a design, including those set by the *set_min_pulse_width* command.

Examples

The following example removes a minimum pulse width requirement previously set for clock CK1.

```
pt_shell> remove_min_pulse_width [get_clocks CK1]
```

The following example removes a minimum pulse width requirement previously set for pin U1/Z.

pt_shell> *remove_min_pulse_width U1/Z*

See Also

- current_design
- report_constraint
- report_min_pulse_width
- reset_design
- set_min_pulse_width

---

remove_multi_scenario_design

Removes all multi-scenario objects from memory and removes from disk all images generated by multi-scenario analysis.

Syntax

status *remove_multi_scenario_design*

Arguments

None.

Description

This command is available only if you invoke the pt_shell with the *-multi_scenario* option.

The *remove_multi_scenario_design* command removes all multi-scenario objects from pt_shell. The current session and all scenarios are removed from memory. All netlist, baseline and current images generated during the multi-scenario analysis are removed from disk.

Feedback

Examples

The following example removes all multi-scenario objects and removes all images from the disk.

```
pt_shell> remove_multi_scenario_design
1
```

See Also

- report_multi_scenario_design

---

remove_net

Removes nets from the current design.

Syntax

int *remove_net*

```
-all
net_list
```

Data Types

```
net_list      list
```

Arguments

`-all`

Indicates that all nets are to be removed from the current design. The *-all* and *net_list* options are mutually exclusive; you can specify only one.

net_list

Specifies a list of nets to be removed from the current design. The *-all* and *net_list* options are mutually exclusive; you can specify only one.

Description

The *remove_net* command removes specified nets or all nets from the current design. Like all other netlist editing commands, for the *remove_net* command to succeed, all of its arguments must succeed. If any argument fails, the netlist remains unchanged. The *remove_net* command returns a 1 if successful and 0 if unsuccessful.

Each net specified must be in scope; that is, at or below the current instance.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported

using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Examples

In the following example, nets are removed if their names begin with "new":

```
pt_shell> remove_net [get_nets new*]
Information: Removed net 'new_net1'. (NED-017)
Information: Removed net 'new_net2'. (NED-017)
1
```

Feedback

See Also

- size_cell
- swap_cell
- write_changes

---

remove_noise_immunity_curve

Removes noise immunity curve for a library pin or port.

Syntax

int *remove_noise_immunity_curve*

```
[-above]
[-below]
[-low]
[-high]
object_list
```

Data Types

```
object_list      list
```

Arguments

`-above`

Removes the noise immunity curve for above ground or power rail noise analysis region.

`-below`

Removes the noise immunity curve for below ground or power rail noise analysis region.

`-low`

Removes the noise immunity curve for ground rail noise.

`-high`

Removes the noise immunity curve for power rail noise. 1.0.

object_list

Specifies a list of lib-pins or ports.

Description

This command removes the noise immunity curve information set by the *set_noise_immunity_curve* command.

Feedback

Examples

This example removes noise immunity curve information for the above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_noise_immunity_curve -above lsi_10k/IV/A
```

See Also

- set_noise_immunity_curve

---

remove_noise_lib_pin

Removes an equivalent noise library pin for a driver or load.

Syntax

int *remove_noise_lib_pin*

pins

Data Types

pins list

Arguments

pins

Specifies the collection of pins for which the equivalent noise library pin is set by the *set_noise_lib_pin* command that needs to be removed.

Description

Given a collection of pins, this command allows you to remove a noise library pin previously set as an equivalent in terms of library noise information.

Examples

The following example removes any equivalent noise information on two input pins of the design, which is the same as an input library pin:

```
pt_shell> remove_noise_lib_pin [get_pins {cell/pin1 cell/pin2}] \\
```

See Also

- set_noise_lib_pin
- update_noise

Feedback

- report_noise
- report_noise_calculation

---

remove_noise_margin

Removes noise margin for a library pin or port.

Syntax

int *remove_noise_margin*

```
[-above]
[-below]
[-low]
[-high]
object_list
```

Data Types

object_list	list

Arguments

`-above`

Removes the noise margin for above ground or power rail noise analysis region.

`-below`

Removes the noise margin for below ground or power rail noise analysis region.

`-low`

Removes the noise margin for ground rail noise.

`-high`

Removes the noise margin for power rail noise. 1.0.

object_list

Specifies a list of lib-pins or ports.

Description

This command removes the noise margin information set by the *set_noise_margin* command.

Feedback

Examples

This example removes noise margin information for above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_noise_margin -above lsi_10k/IV/A
```

See Also

- set_noise_margin

---

remove_ocvm

Removes advanced and parametric on-chip variation (AOCV / POCV) information.

Syntax

status *remove_ocvm*

```
[-coefficient]
[-derate]
[object_list]
```

Data Types

object_list list

Arguments

`-coefficient`

Removes only the AOCV coefficients. The *-coefficient* and *-derate* options are mutually exclusive.

`-derate`

Removes only the AOCV/POCV derating factors. The *-coefficient* and *-derate* options are mutually exclusive.

object_list

Specifies the objects from which AOCV/POCV information is removed.

Description

Removes user-specified AOCV coefficients and AOCV/POCV derating factors that were previously set with the *set_aocvm_coefficient* and *read_ocvm* commands, respectively. If the command is specified with no options, all AOCV/POCV data is removed.

To display AOCV/POCV deratings and coefficients, use the *report_ocvm* command.

Feedback

Examples

The following example removes the AOCV coefficients from the library cell named lib/bufA:

```
pt_shell> remove_ocvm -coefficient lib/bufA
```

The following example removes all AOCV/POCV derating factors that were set using the *read_ocvm* command:

```
pt_shell> remove_ocvm -derate
```

The following example removes all AOCV/POCV derating factors from the current design:

```
pt_shell> remove_ocvm -derate [current_design]
```

See Also

- read_ocvm
- report_ocvm
- set_aocvm_coefficient

---

remove_operating_conditions

Removes operating conditions from current design, cells, or ports.

Syntax

string *remove_operating_conditions*

```
[-object_list objects]
```

Data Types

objects list

Arguments

-object_list objects

Specifies the cells or ports to remove operating conditions from. The command undoes operating conditions set by the *set_operating_conditions* command. Without the *-object_list* option, all the operating conditions are removed from the design. Both leaf cells and hierarchical blocks are accepted with the *-object-list* option.

Description

Removes operating conditions settings from the current design, individual blocks, leaf cells, or ports.

Feedback

Examples

This example removes all operating conditions from the current design. Both design-specific and cell-specific operating conditions are removed.

```
pt_shell> remove_operating_conditions
```

See Also

- set_operating_conditions

---

remove_output_delay

Removes output delay from output ports or pins.

Syntax

string *remove_output_delay*

```
[-clock clock_name]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-sms_scenarios sms_scenarios_list]
port_pin_list
```

Data Types

clock_name	string
port_pin_list	list
sms_scenarios_list	collection

Arguments

`-clock` *clock_name*

Relative clock; {""} for input delay relative to no clock.

`-clock_fall`

Removes the delay relative to falling edge of clock. If you specify the *clock_name* variable without the *-clock_fall* command, the delay relative to rising edge of the clock is removed.

`-level_sensitive`

Removes level-sensitive output delay.

`-rise`

Removes rising output delay.

`-fall`

Removes falling output delay.

`-max`

Removes maximum output delay.

`-min`

Removes minimum output delay.

`port_pin_list`

Specifies a list of ports and pins. Each element in the list is either a collection of ports or pins, or a pattern which matches ports or pins on the current design.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only output delays compatible with the specified SMS scenarios. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

Removes output delay values for objects in the current design. Set output delay with the *set_output_delay* command. By default, all output delays on each object in the *port_pin_list* command are removed. To restrict the removed output delay values, use the *-clock*, *-clock_fall*, *-min*, *-max*, *-rise*, or *-fall* options. To show output delays associated with ports, use the *report_port -output_delay* command.

Examples

The following example removes all output delay values from all output ports in the current design.

```
pt_shell> remove_output_delay [all_outputs]
```

The following example removes output maximum rise delay values from OUT1, OUT3, and BIDIR12.

```
pt_shell> remove_output_delay -max -rise {OUT1 OUT3 BIDIR12}
```

The following example removes all output delays for port OUT1 relative to the falling edge of CLK2.

```
pt_shell> remove_output_delay -clock CLK2 -clock_fall OUT1
```

The following example removes all output delays not relative to any clock for port OUT2.

```
pt_shell> remove_output_delay -clock {""} OUT2
```

See Also

- all_outputs
- collections
- report_port
- set_output_delay
- set_input_delay

---

remove_parasitic_corner

Removes a previously set parasitic corner in the presence of variation-aware parasitics.

Syntax

status *remove_parasitic_corner*

Arguments

None.

Description

The *remove_parasitic_corner* command removes the parasitic corner information previously set via the *set_parasitic_corner* command. The command succeeds only if variation-aware parasitics were annotated and the *set_parasitic_corner* command is used to set the parasitic corner.

After using this command, all the parasitic attributes and analysis use nominal values instead of corner values.

Examples

The following example removes the parasitics corner information set previously.

```
pt_shell> remove_parasitic_corner
```

See Also

- set_parasitic_corner
- read_parasitics
- report_annotated_parasitics

Feedback

remove_path_group

Removes path group objects.

Syntax

string *remove_path_group*

```
-all
path_group_list
```

Data Types

```
path_group_list        list
```

Arguments

`-all`

Removes all path groups in the current design.

`path_group_list`

Specifies path group names to remove. Each element in the list is either a collection of path groups or a pattern matching path group names.

Description

Removes path groups in the current design. The *group_path* and *create_clock* commands create path groups. Path groups affect the cost function for optimization and influence the timing reports. If you remove a path group, any paths in that group are implicitly assigned to the default path group. To show path group information, use the *report_path_group* command.

Examples

The following example removes all path groups in the design.

```
pt_shell> remove_path_group -all
```

The following example removes path group "BUS1".

```
pt_shell> remove_path_group BUS1
```

See Also

- collections
- create_clock
- group_path

Feedback

- report_path_group
- report_timing

---

remove_path_tag_set

Removes path tag sets or removes paths from path tag sets.

Syntax

status *remove_path_tag_set*

```
[path_collection]
[-name tag_names]
```

Data Types

```
path_collection     collection
tag_names           list
```

Arguments

path_collection

Specifies a collection of paths. The command removes these paths from the named path tag sets or all path tag sets, reducing the number of paths in those path tag sets. Without this option, the command removes whole path tag sets.

-name *tag_names*

Specifies a single path tag or a list of path tag sets to remove, or to reduce by removing the paths specified by the *path_collection* option. Without the *-name* option, the command applies to all path tag sets.

Description

This command removes a named path tag set or list of named path tag sets, or all path tag sets if the *-name* option is not used.

If you specify a path collection, the command removes those paths from the named path tag sets or all path tag sets.

Examples

The following example removes all path tag sets.

```
pt_shell> remove_path_tag_set
```

The following example removes the path tag sets named tag1 and tag3.

```
pt_shell> remove_path_tag_set -name {tag1 tag3}
```

The following example removes a collection of paths from the tag1 path set.

```
pt_shell> remove_path_tag_set -name tag1 [get_timing_paths -max_paths 20]
```

See Also

- create_path_tag_set
- report_path_tag_set
- enable_path_tagging

---

remove_placement_blockage

Removes a placement blockage.

Syntax

status *remove_placement_blockage*

```
[-name blockage_name]
[-all]
```

Data Types

```
blockage_name     string
```

Arguments

`-name blockage_name`

Specifies the optional name of the blockage.

Description

This command removes a placement blockage that controls the placement of ECO cells when you use the *fix_eco_drc* or *fix_eco_timing* command with the *-physical_mode* option.

You can remove a placement blockage to remove the placement blockages created by *create_placement_blockage* command. You can also specify the -all option to remove all placement blockages.

Examples

The following example removes a placement blockage named placeblockage_10.

```
pt_shell> remove_placement_blockage -name placeblockage_10
```

Feedback

See Also

- create_placement_blockage
- write_changes

---

remove_port_fanout_number

Removes fanout number information on ports.

Syntax

string *remove_port_fanout_number*

```
        port_list
```

Data Types

port_list	list

Arguments

port_list

Specifies a list of ports. Each element in the list is either a collection of ports or a pattern that matches ports on the current design.

Description

Removes the *fanout_number* settings on ports. The *set_port_fanout_number* command sets the number of external fanout pins on ports, which is used along with wire load models to calculate capacitance and resistance of nets. To show port fanout number information, use the *report_port -wire_load* command.

Examples

This example removes the port fanout_number settings from ports OUT1*.

```
pt_shell> remove_port_fanout_number [get_ports OUT1*]
```

See Also

- collections
- get_ports
- report_port

Feedback

- set_port_fanout_number
- set_wire_load_model

---

remove_power_groups

Remove the existing power groups.

Syntax

string *remove_power_groups*

```
-all
group_names
```

Data Types

```
group_names          list
```

Arguments

`-all`

Removes all the user-defined power groups. Note that predefined power groups are note deleted using this option.

group_names

Specifies the groups to be removed.

Description

The *remove_power_groups* command removes some or all power groups that are previously created but no longer interested. The predefined power groups can be removed, but not with the *-all* option. The predefined power group names need to be explicitly specified.

Examples

In the following example, all the user defined power groups are removed.

```
pt_shell> remove_power_groups -all
1
```

In the following example, the predefined clock_network power group is removed.

```
pt_shell> remove_power_groups clock_network
1
```

Feedback

See Also

- create_power_group
- report_power_groups
- get_power_group_objects

---

remove_propagated_clock

Removes a propagated clock specification.

Syntax

string *remove_propagated_clock*

object_list

Data Types

object_list list

Arguments

object_list

Lists clocks, ports, or pins.

Description

Removes propagated clock specification from clocks, ports, or pins in the current design. The *set_propagated_clock* command specifies that delays be propagated through the clock network to determine latency at register clock pins. If this is not specified, ideal clocking is assumed. Ideal clocking means clock networks have a user specified latency (set by the *set_clock_latency* command) or zero latency, by default. Propagated clock latency is normally used for post layout, after final clock tree generation.

Ideal clock latency provides an estimate of the clock tree for prelayout. You can also use the *set_clock_latency* command to specify an ideal latency, which overrides the propagated clock specification for an object.

For information about propagated clock attributes on clocks, see the *report_clock* man page.

Examples

The following example removes propagated clock specifications from all clocks in the design.

```
pt_shell> remove_propagated_clock [all_clocks]
```

Feedback

See Also

- report_clock
- set_clock_latency
- set_propagated_clock

---

remove_pulse_clock_max_transition

Removes maximum transition limits from pulse clock network and input of pulse generator.

Syntax

string *remove_pulse_clock_max_transition*

```
[-rise]
[-fall]
[-transitive_fanout]
object_list
```

Data Types

```
object_list       list
```

Arguments

`-rise`

Removes the rise transition constraint.

`-fall`

Removes falling transition constraint.

`-transitive_fanout`

Removes the constraint from the transitive fanout of the pulse generator. If this option is not specified, the constraint is removed from the input of pulse generator.

`object_list`

Lists the pulse generator cell, pulse generator lib cell, clock, and design from which to remove maximum pulse clock transition limits.

Description

Removes maximum pulse clock transition limits from the specified objects. A maximum transition limit is specified with the *set_pulse_clock_max_transition* command. This command can change the constraint on the object, which has conflicting constraints due to overlap. You can use the *-rise* and *-fall* options to remove only the rising or the falling transition respectively. If neither the *-rise* nor the *-fall* option is specified, both rise and fall

transition limits are removed. You must specify the *-transitive_fanout* option to remove pulse clock maximum transition constraint from the transitive fanout of pulse generators.

Examples

The following example removes the pulse clock maximum transition limit from the input of all the cells corresponding to the library cell pulse_rise_high in the library celllib.

```
pt_shell> remove_pulse_clock_max_transition {"celllib/pulse_rise_high"}
```

The following example removes maximum transition limit from all the pulse clock networks in the clock network clk

```
pt_shell> remove_pulse_clock_max_transition -transitive_fanout
 [get_clocks clk]
```

See Also

- current_design
- set_pulse_clock_max_transition
- report_pulse_clock_max_transition
- report_constraint

---

remove_pulse_clock_max_width

Removes maximum pulse width limits from pulse clock network.

Syntax

string *remove_pulse_clock_max_width*

```
[-transitive_fanout]
object_list
```

Data Types

object_list	list

Arguments

`-transitive_fanout`

Remove the constraints from the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

object_list

Lists the pulse generator cell, pulse generator lib cell, clock and design from which to remove maximum pulse clock width limits.

Feedback

Description

Removes maximum pulse clock width limits from the specified objects. A maximum width limit is specified with the *set_pulse_clock_max_width* command. This command can change the constraint on the object, which has conflicting constraints due to overlap.

Examples

The following example removes maximum width limit from all the pulse clock networks in the design.

```
pt_shell> remove_pulse_clock_max_width [current_design]
```

See Also

- current_design
- set_pulse_clock_max_width
- report_pulse_clock_max_width
- report_constraint

---

remove_pulse_clock_min_transition

Removes minimum transition limits from input of pulse generator.

Syntax

string *remove_pulse_clock_min_transition*

```
[-rise]
[-fall]
object_list
```

Data Types

object_list	list

Arguments

`-rise`

Removes the rise transition constraint.

`-fall`

Removes the falling transition constraint.

object_list

Lists the pulse generator cell, pulse generator lib cell, clock and design from which to remove minimum pulse clock transition limits.

Feedback

Description

Removes minimum pulse clock transition limits from the input of specified objects. If clock or design is specified, the minimum transition limit is removed from the input of all the pulse generators in the clock network or design, respectively. A minimum transition limit is specified with the *set_pulse_clock_min_transition* command. This command can change the constraint on the object, which has conflicting constraints due to overlap. You can use the *-rise* and *-fall* options to remove only the rising or the falling transition, respectively. If neither the *-rise* or *-fall* option is specified, both rise and fall transition limits are removed.

Examples

The following example removes a minimum transition limit from input of all the cells corresponding to the *pulse_rise_high* library cell in the *celllib* library.

```
pt_shell> remove_pulse_clock_min_transition {"celllib/pulse_rise_high"}
```

See Also

- current_design
- set_pulse_clock_min_transition
- report_pulse_clock_min_transition
- report_constraint

---

remove_pulse_clock_min_width

Removes minimum pulse width limits from pulse clock network.

Syntax

string *remove_pulse_clock_min_width*

```
[-transitive_fanout]
object_list
```

Data Types

object_list	list

Arguments

`-transitive_fanout`

Removes the constraints from the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

object_list

Lists the pulse generator cell, pulse generator lib cell, clock and design from which to remove minimum pulse clock width limits.

Description

Removes minimum pulse clock width limits from the specified objects. A minimum width limit is specified with the *set_pulse_clock_min_width* command. This command can change the constraint on the object, which has conflicting constraints due to overlap.

Examples

The following example removes minimum width limit from all the pulse clock networks in the clock network clk.

```
pt_shell> remove_pulse_clock_min_width [get_clocks clk]
```

See Also

- current_design
- set_pulse_clock_min_width
- report_pulse_clock_min_width
- report_constraint

---

remove_qtm_attribute

Removes an attribute setting from the QTM object.

Syntax

string *remove_qtm_attribute*

```
-class lib | lib_cell | lib_pin
attribute_name
object_names
```

Data Types

```
attribute_name      string
object_names        list
```

Arguments

```
-class lib | lib_cell | lib_pin
```

Specifies the class of the QTM objects to remove attribute from.

attribute_name

Specifies the name of the attribute.

object_names

Specifies the names of the objects from which to remove the named attribute. Each element in the list must be a name to identify an object in the specified class. The object names should not be specified when the object class is either *lib* or *lib_cell*. It must be specified when the class is *lib_pin*, in which case the object names should be the names of ports already defined for the QTM.

Description

The *remove_qtm_attribute* command removes attributes you set with the *set_qtm_attribute* command.

You can remove either application attributes or user attributes you defined for QTM object with the *define_qtm_attribute* command.

Examples

The following example defines the is_XYZ Boolean attribute for QTM cell, then sets the value on the QTM cell under construction. Finally, it removes the attribute from the cell.

```
pt_shell> define_qtm_attribute -type boolean -class lib_cell "is_XYZ"
pt_shell> set_qtm_attribute -class lib_cell "is_XYZ" "true"
pt_shell> remove_qtm_attribute -class lib_cell "is_XYZ"
```

The following example defines the my_float_attr user attribute for all QTM port objects, sets the value for two QTM ports, and removes the attribute from one of the ports.

```
pt_shell> define_qtm_attribute -type float -class lib_pin
 "my_float_attr"
pt_shell> set_qtm_attribute -class lib_pin "my_float_attr" 100.0 {IN,
 OUT}
pt_shell> remove_qtm_attribute -class lib_pin "my_float_attr" {OUT}
```

See Also

- define_qtm_attribute
- define_user_attribute
- set_qtm_attribute

---

remove_qtm_constraint_arc

Removes a constraint arc for a quick timing model.

Feedback

Syntax

string *remove_qtm_constraint_arc*

```
[-name arc_names]
[-setup]
[-hold]
[-recovery]
[-removal]
[-from port_spec]
[-to port_spec]
[-edge rise | fall]
[-signal_edge rise | fall]
[-quiet]
```

Data Types

arc_names	list
port_spec	list

Arguments

`-name arc_names`

Specifies the names of constraint arcs to be removed.

`-setup`

Removes setup arcs.

`-hold`

Removes hold arcs.

`-recovery`

Removes recovery arcs.

`-removal`

Removes removal arcs.

`-from port_spec`

Remove arcs with given starting constraining clock port.

`-to port_spec`

Remove arcs with given ending constrained input/inout port.

`-edge rise | fall`

Remove arcs with given triggering edge.

`-signal_edge rise | fall`

Remove arcs with given direction of the signal at the constrained port.

Feedback

`-quiet`

Suppress the warning message and silently return true if no arc is removed.

Description

This command allows you to remove previously defined constraint arcs by *create_qtm_constraint_arc* in a quick timing model. The options are filters to select matching QTM arcs. More options make the removal more specific. The *-from* port denotes the constraining port, and the *-to* port denotes the constrained port. The *-from* port must be defined as a clock port and the *-to* port must be an input or inout port. You can use *-signal_edge* option to specify the the direction of the signal at the constrained port specified by the -to option.

Use the *-setup*, *-hold*, *-recovery* and *-removal* options to remove setup, hold, recovery or removal arcs. You must use at leaset one of the switches. Use the *-edge* option and specify rise or fall for the triggering clock of the arcs to be removed.

If no predefined QTM arcs satisfy the condition in the command, no arcs will be removed. A warning will be issued unless you use the *-quiet* option.

Examples

The following commands create and remove setup arcs from the positive edge of constraining port CLK to IN1, and then create a new arc with updated constraint value.

```
pt_shell> create_qtm_constraint_arc -setup -edge rise -from CLK -to IN1
 -value 3.0
pt_shell> remove_qtm_constraint_arc -setup -edge rise -from CLK -to IN1
pt_shell> create_qtm_constraint_arc -setup -edge rise -from CLK -to IN1
 -value 5.0
```

The following command removes all setup and hold arcs from the positive edge of constraining port CLK.

```
pt_shell> create_qtm_constraint_arc -setup -hold -edge rise -from CLK
```

See Also

- create_qtm_constraint_arc
- create_qtm_delay_arc
- create_qtm_model
- create_qtm_path_type
- get_qtm_ports
- report_qtm_model
- save_qtm_model

Feedback

remove_qtm_delay_arc

Removes a delay arc for a quick timing model (QTM).

Syntax

string *remove_qtm_delay_arc*

```
[-name arc_names]
[-from port_spec
[-to port_spec]
[-edge rise | fall]
[-from_edge rise | fall]
[-to_edge rise | fall]
[-quiet]
```

Data Types

arc_names	list
port_spec	list

Arguments

`-name arc_names`

Specifies the names of delay arcs to be removed.

`-from port_spec`

Specifies the startpoint of the delay arc to be removed. This must be QTM port name or a collection of QTM ports.

`-to port_spec`

Specifies the endpoint of the delay arc to be removed. This must be QTM port name or a collection of QTM ports with output or inout type.

`-edge rise | fall`

Remove arcs with given triggering edge.

`-from_edge rise | fall`

Remove arcs with given triggering edge (clock startpoints) or the signal direction (data startpoints).

`-to_edge rise | fall`

Remove arcs with given the signal direction at the data endpoints.

`-path_type path_type`

Specifies the path type you want to use to set the delay.

`-quiet`

Suppress the warning message and silently return true if no arc is removed.

Description

This command allows you to remove previously defined delay arcs by *create_qtm_delay_arc* in a quick timing model.

The options are filters to select matching QTM arcs. More options make the removal more specific. By default, if you specify a list of ports for the -from and *-to* ports, all delay arcs between these ports will be removed.

One of *-from* and *-to* option is required for the command. To remove only edge triggered arcs, use the -edge option.You can use *-from_edge* and *-to_edge* to remove arcs with specify the triggered edge for clock start points or the signal direction for data startpoints and endpoints.

If no predefined QTM arcs satisfy the condition in the command, no arcs will be removed. A warning will be issued unless you use the *-quiet* option.

Examples

The following commands create and removes a delay arc from input ports IN* to output port OUT with a delay value of 2.0 time units, and then creates a new arc with updated delay values.

```
pt_shell> create_qtm_delay_arc -from [get_qtm_ports IN*] -to OUT -edge
 rise -value 2.0
pt_shell> remove_qtm_delay_arc -from [get_qtm_ports IN*] -to OUT -edge
 rise
pt_shell> create_qtm_delay_arc -from [get_qtm_ports IN*] -to OUT -edge
 rise -value 5.0
```

The following commands first create and remove a delay arc from clock CLK to output ports OUT* with a delay value of 2.0 time units. And then it create a new generated clock and creates a new delay arc starting from it.

```
pt_shell> create_qtm_delay_arc -from CLK -to [get_qtm_ports OUT*] -edge
 rise -value 2.0
pt_shell> remove_qtm_delay_arc -from CLK -to [get_qtm_ports OUT*] -edge
 rise
pt_shell> create_qtm_generated_clock GCLK -source CLK -divide_by 2
pt_shell> create_qtm_delay_arc -from GCLK -to [get_qtm_ports OUT*] -edge
 rise -value 5.0
```

See Also

- create_qtm_delay_arc
- create_qtm_constraint_arc

Feedback

- create_qtm_model
- create_qtm_path_type
- get_qtm_ports
- report_qtm_model
- save_qtm_model

---

remove_qtm_generated_clock

Removes a QTM generated clock.

Syntax

string *remove_qtm_generated_clock*

```
[clock_list]
[-all]
[-quiet]
```

Data Types

```
clock_list        list
all            boolean
quiet          boolean
```

Arguments

clock_list

Specifies a list of names of the generated clocks to be removed.

-all

Remove all generated clocks in the qtm model. *clock_list* and *-all* are mutually exclusive. At least one of these options must be given.

-quiet

Suppress the warning message and silently return true if no clock is removed.

Description

This command removes QTM generated clocks created by *create_qtm_generated_clock*. When removing a QTM generated clock, the command will automatically remove all arcs triggered by the clock. All generated clocks whose master clock is this clock will also be removed.

If the generated clock to be removed is defined on an internal port in the QTM, that internal port will also be removed.

Feedback

If no predefined QTM generated clocks satisfy the condition in the command, no clocks will be removed. A warning will be issued unless you use the *-quiet* option.

Examples

The following example creates and removes a divide by 2 generated clock on an internal port, assume there is no port defined with name "GCLK_int", with master clock CLK. It then creates a new divide by 3 generated clock to replace it.

```
pt_shell> create_qtm_generated_clock GCLK_int -source CLK -divide_by 2
pt_shell> remove_qtm_generated_clock GCLK_int
pt_shell> create_qtm_generated_clock GCLK_int -source CLK -divide_by 3
```

The following example removes all generated clocks in the QTM.

```
pt_shell> remove_qtm_generated_clock -all
```

See Also

- create_qtm_generated_clock
- create_qtm_model
- get_qtm_ports
- report_qtm_model
- save_qtm_model

---

remove_qtm_port

Removes a quick timing model (QTM) port.

Syntax

string *remove_qtm_port*

```
port_list
[-quiet]
```

Data Types

```
port_list     list
```

Arguments

port_list

Specifies the list of QTM ports you want removed.

Feedback

`-quiet`

Suppress the warning message and silently return true if no port is removed.

Description

This command removes QTM ports created by *create_qtm_port*. You can remove a single port or a list of ports with this command. When removing a QTM port, the command will automatically remove all arcs connected to the ports. If the port is a clock port, all generated clocks whose master clock is this will also be removed.

Bus ports must be removed together by giving the start and end index (A[0:5]). A single bus port can not be removed.

If no predefined QTM ports satisfy the condition in the command, no ports will be removed. A warning will be issued unless you use the *-quiet* option.

Examples

The following example removes a bused QTM port A[0:5].

```
pt_shell> remove_qtm_port A[0:5]
```

The following example removes QTM ports A, B, C, and D.

```
pt_shell> remove_qtm_port {A B C D}
```

See Also

- create_qtm_port
- create_qtm_model
- get_qtm_ports
- report_qtm_model
- save_qtm_model
- set_qtm_port_drive
- set_qtm_port_load

---

remove_rail_voltage

Removes power rail voltage that was set by the *set_rail_voltage* command on cells.

Syntax

int *remove_rail_voltage*

```
        cell_list
```

Data Types

```
cell_list          list
```

Arguments

cell_list

Specifies a list of cells from which to remove rail voltages.

Description

Removes power rail voltage that was set by the *set_rail_voltage* command on cells. If dynamic components of rail voltage have been specified they are also removed along with the total rail voltage. This command "undoes" the voltages specified by the *set_rail_voltage* command. It can be applied to both leaf cells and hierarchical blocks.

Examples

The following example removes rail voltages from a cell named *h1/u3*.

```
pt_shell> remove_rail_voltage [get_cells {h1/u3}]
1
```

See Also

- set_operating_conditions
- set_rail_voltage

---

remove_resistance

Removes resistance on nets.

Syntax

status *remove_resistance*

```
        net_list
```

Data Types

```
net_list     list
```

Feedback

Arguments

net_list

Specifies a list of nets in the current design, whose resistances are removed.

Description

Specifies that the lumped resistance annotated on the list of nets must be removed. PrimeTime will revert to using the resistance from detailed parasitics (set using the *read_parasitics* command), if they exist. If not, the internally estimated net resistance is used.

Annotated resistances can also be removed on the full design by using the *reset_design* command.

Examples

The following example removes the annotated resistance on net w23.

```
pt_shell> remove_resistance [get_nets "w23"]
```

See Also

- all_outputs
- current_design
- set_resistance
- report_net
- report_port
- reset_design
- set_drive

---

remove_ruleset

Removes specified rulesets.

Syntax

Boolean *remove_ruleset*

ruleset_list

```
list ruleset_list
```

Feedback

Arguments

ruleset_list

The list of rulesets to remove.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Removes specified rulesets. Note that removing a ruleset does not remove the rules in that ruleset, as one rule may be in multiple rulesets. To remove all rules in a ruleset, use the *remove_rule* command instead.

Examples

The following example removes the ruleset named "my_set1".

```
pt_shell> remove_ruleset my_set1
```

See Also

- create_rule
- create_ruleset
- get_rules

---

remove_scenario

Removes a scenario in the multi-scenario analysis mode.

Syntax

status *remove_scenario*

```
        scenario_list
```

Data Types

scenario_list list

Arguments

scenario_list

Specifies unique strings used to identify each scenario.

Description

This command is available only if you invoke the pt_shell with the *-multi_scenario* option.

The *remove_scenario* command removes the specified scenario from memory. To determine the current scenarios that exist, execute the following:

```
report_multi_scenario_design -scenarios
1
```

Examples

In the following example, two scenarios named scen1 and scen2 are removed.

```
pt_shell> remove_scenario {scen1 scen2}
1
```

See Also

- create_scenario
- report_multi_scenario_design

---

remove_scenario_case_analysis

Removes the scenario case analysis value on a port or pin.

Syntax

```
status remove_scenario_case_analysis
  port_or_pin_list
  [-sms_scenarios sms_scenarios_list]
  [-all]
```

Data Types

```
port_or_pin_list     list
sms_scenarios_list           collection
```

Arguments

port_or_pin_list

Removes case analysis from the specified ports or pins.

-sms_scenarios

Describes the scenarios on which the scenario case analysis is to be removed. If this option is ommitted, all scenario case analysis is removed from the ports or pins.

all

Removes all scenario case analysis throughout the design.

Feedback

Description

This command removes previously specified entries of scenario case analysis on ports or pins. You must specify either the *port_or_pin_list* option, or *-all*.

Regular case analysis set using *set_case_analysis* is not removed by *remove_scenario_case_analysis*.

See Also

- report_case_analysis
- set_scenario_case_analysis

---

remove_sense

Removes sense information defined on pins or cell timing arcs.

Syntax

```
status remove_sense
   [-type clock | data]
   [-clocks clock_list]
   [-all]
   object_list
```

Data Types

```
clock_list      list
object_list     list
```

Arguments

`-type clock | data`

Specifies the removal of sense from *clock* (the default) or *data* networks.

`-clocks clock_list`

Specifies a list of clock objects to be associated with the given pin objects in the *object_list*. This option removes the unateness only for the specified clock domain. This option can only remove clock sense predefined by the *set_sense -type clock* command. It does not remove the default clock sense setting for this given pin.

`-all`

Removes all unateness information in current design. Please note that when no type is specified, it removes all the sense information from *clock* (the default) network.In order to remove all the sense information from the *data* network, please explicitly specify *-type data*.

object_list

Specifies a list of pins or cell timing arcs from which to remove predefined unateness.

Description

This command removes the unateness information that was previously defined by the *set_sense* command.

Examples

This example removes positive unateness defined on the XOR/Z pin with respect to the CLK1 clock but does not remove the negative unateness.

```
pt_shell> set_sense -positive -clocks [get_clocks CLK1] XOR/Z
pt_shell> set_sense -negative XOR/Z
pt_shell> remove_sense -clocks [get_clocks CLK1] XOR/Z
```

This example removes all unateness information in the current design.

```
pt_shell> remove_sense -all
```

See Also

- set_sense

---

remove_si_aggressor_exclusion

Removes the exclusive groups set by the *set_si_aggressor_exclusion* command.

Syntax

status *remove_si_aggressor_exclusion*

```
[-rise]
[-fall]
[-all]
[nets]
```

Data Types

nets list

Arguments

-rise

Removes the exclusive group set for the aggressor nets *anets* in the *rise* direction. If neither the *-rise* nor the *-fall* option is specified, the exclusive groups of the aggressor nets *anets* in both *-rise* and *-fall* directions are removed.

`-fall`

Removes the exclusive group set for the aggressor nets *anets* in the *fall* direction. If neither the *-rise* nor the *-fall* option is specified, the exclusive groups of the aggressor nets *anets* in both *-rise* and *-fall* directions are removed.

`-all`

Removes all the exclusive groups set on the design.

nets

Specifies the list of nets belonging to one exclusive group that are to be removed.

Description

The *remove_si_aggressor_exclusion* command removes the effect of the *set_si_aggressor_exclusion* command that was set on the aggressor nets. Only those groups that have been set can be removed. A warning message is issued if you are trying to remove a group that has not been set.

These exclusive commands are independent of parasitics, so they can be applied even before reading in parasitics.

Note that application of exclusive aggressors are not done incrementally. The next *update_timing* and *update_noise* commands would produce a full update.

You can use the *report_si_aggressor_exclusion* command to check whether the *remove_si_aggressor_exclusion* command was applied to the exclusive groups.

Examples

The following example shows how to remove the exclusion on nets *SCAN_LOGIC** for rise direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets SCAN_LOGIC*] -rise
1
pt_shell> remove_si_aggressor_exclusion [get_nets SCAN_LOGIC*]
1
```

The following example shows how to remove all exclusive groups set on the design.

```
pt_shell> remove_si_aggressor_exclusion -all
1
```

See Also

- report_delay_calculation
- report_noise_calculation
- report_si_aggressor_exclusion

Feedback

- set_si_aggressor_exclusion
- set_si_delay_analysis
- set_si_noise_analysis
- si_analysis_logical_correlation_mode

---

remove_si_delay_analysis

Removes the effect of the *set_si_delay_analysis* command.

Syntax

int *remove_si_delay_analysis*

```
[-ignore_arrival inets]
[-victims vnets]
[-aggressors anets]
[-rise]
[-fall]
[-min]
[-max]
[-all]
```

Data Types

```
inets        list
vnets        list
anets        list
```

Arguments

`-ignore_arrival inets`

Removes the effect of using the *set_si_delay_analysis* command with its *-ignore_arrival* option. You cannot use this option with the *-victims* and *-aggressors* options.

`-victims vnets`

Removes the effect of using the *set_si_delay_analysis* command with its *-victims* option. When you use the *-victims* option with the *-aggressors* option, the command restores the pair-wise relationship. However, when the *-victims* option is used to remove the effect set by the *set_si_delay_analysis* command with the *-aggressors* option, it does not remove any exclusion on the nets. You cannot use the *-victims* option with the *-ignore_arrival* option.

`-aggressors anets`

Removes the effect of the *set_si_delay_analysis* command with its *-aggressors* option. When you use the *-aggressors* option with the *-victims* option, the

command restores the pair-wise relationship. However, when the *-aggressors* option is used to remove the effect set by the *set_si_delay_analysis* command with the o-*victims* option, it does not remove any exclusion on the nets. You cannot use the *-aggressors* option with the *-ignore_arrival* option.

`-rise`

Removes the effect of the *set_si_delay_analysis -exclude -rise* command.

`-fall`

Removes the effect of the *set_si_delay_analysis -exclude -fall* command.

`-min`

Removes the effect of the *set_si_delay_analysis -exclude -min* command.

`-max`

Removes the effect of the *set_si_delay_analysis -exclude -max* command.

`-all`

Removes all the effects of the *set_si_delay_analysis* command on all the nets.

Description

Removes the effect of the *set_si_delay_analysis* command.

The *set_si_delay_analysis* command allows you to exclude nets from crosstalk analysis or setting some net as infinite window as infinite window as both aggressor and victim . The *remove_si_delay_analysis* command removes such overrides.

The *remove_si_delay_analysis* command returns a *1* if successful and a *0* if unsuccessful. If the remove command is used to remove a effect that has not been set, the warning message *XTALK-107* is issued.

To view the result of this command, use the *report_si_delay_analysis* command.

Examples

In the following example, all the nets named *CLK_NET_** are returned to their original state in crosstalk analysis.

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets CLK_NET_*]
1

pt_shell> remove_si_delay_analysis -victims [get_nets CLK_NET_*]
1
```

However in the following examples, none of the nets named *REG_NET_** are returned to their original state in crosstalk analysis.

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets REG_NET_*]
1

pt_shell> remove_si_delay_analysis -aggressors [get_nets REG_NET_*]
Warning: Cannot remove an effect that was not set on net(s) REG_NET_1.
 (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s) REG_NET_2.
 (XTALK-107)
1

pt_shell> remove_si_delay_analysis -aggressors [get_nets REG_NET_*]
-victims [get_nets CLK_NET]
1
```

If there is a VIC_NET with aggressors AGG_NET_* and you had set pairwise exclusion on the nets with the list of its aggressors, the AGG_NET_* nets cannot be returned to their original state by using only *-aggressors* option. For example,

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets VIC_NET]
-aggressors [get_nets AGG_NET*]
1

pt_shell> remove_si_delay_analysis -aggressors [get_nets AGG_NET*]
1
```

In order to reset the pairwise exclusion on the nets VIC_NET and AGG_NET_*, the pairwise reset should be used as shown below

```
pt_shell> remove_si_delay_analysis -victims [get_nets VIC_NET]
-aggressors [get_nets AGG_NET*]
1
```

To verify if the exclusion was removed on the nets, use the *report_si_delay_analysis* command with the *-excluded* option.

See Also

- read_parasitics
- set_si_delay_analysis
- report_si_delay_analysis
- si_enable_analysis

---

remove_si_delay_disable_statistical

Removes the effect of the *set_si_delay_disable_statistical* command.

Feedback

Syntax

int *remove_si_delay_disable_statistical*

```
        dnets
```

Data Types

dnets	list

Arguments

dnets

A list of nets for which the effect of the *set_si_delay_disable_statistical* command is removed.

Description

Removes the effect of the *set_si_delay_disable_statistical* command.

The *set_si_delay_disable_statistical* command allows you to disable the statistical analysis for the *dnets* option if selected in composite aggressor group for crosstalk analysis.

The *remove_si_delay_disable_statistical* command removes such effect.

Examples

The following example shows how to put net back into statistical analysis when considered as a composite aggressor.

```
pt_shell> remove_si_delay_disable_statistical [get_nets LOGIC1]
1
```

See Also

- set_si_delay_disable_statistical
- report_si_delay_analysis

---

remove_si_noise_analysis

Removes the effect of the *set_si_noise_analysis* command.

Syntax

int *remove_si_noise_analysis*

```
[-ignore_arrival inets]
[-victims vnets]
```

Feedback

```
[-aggressors anets]
[-above]
[-below]
[-low]
[-high]
[-all]
```

Data Types

inets list *vnets* list *anets* list

Arguments

`-ignore_arrival inets`

Removes the effect of using the *set_si_noise_analysis* command with its *-ignore_arrival* option. You cannot use this option with the *-victims* or *-aggressors* option.

`-victims vnets`

Removes the effect of using the *set_si_noise_analysis* command with its *-victims* option. When the *-victims* option is used to remove the effect set by the *set_si_noise_analysis* command with the *-aggressors* option, it does not remove any exclusion on the nets. However, when you use the *-victims* option with *-aggressors* option, the command restores the pair-wise relationship. You cannot use the *-victims* option with the *I-ignore_arrival* option.

`-aggressors anets`

Removes the effect of the *set_si_noise_analysis* command with its *-aggressors* option. When the *-aggressors* option is used to remove the effect set by the *set_si_noise_analysis* command with the *-victims* option, it does not remove any exclusion on the nets. However, when you use the *-aggressors* option with the *-victims* options, the command restores the pair-wise relationship. You cannot use the *-aggressors* option with the *-ignore_arrival* option.

`-above`

Removes the effect of the *set_si_noise_analysis -exclude -above* command.

`-below`

Removes the effect of the *set_si_noise_analysis -exclude -below* command.

`-low`

Removes the effect of the *set_si_noise_analysis -exclude -low* command.

`-high`

Removes the effect of the *set_si_noise_analysis -exclude -high* command.

`-all`

Removes all the effects of the *set_si_noise_analysis* command on all the nets.

Description

Removes the effect of the *set_si_noise_analysis* command.

The *set_si_noise_analysis* command allows you to exclude nets from noise analysis, or set some net as infinite window as aggressor. The *remove_si_noise_analysis* command removes such overrides.

The *remove_si_noise_analysis* command returns a *1* if successful and a *0* if unsuccessful. If remove command is used to remove a effect that has not been set, the warning message XTALK-107 is issued.

Examples

In the following example, all the nets named *CLK_NET_** are returned to their original state in noise analysis.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets CLK_NET]
1

pt_shell> remove_si_noise_analysis -victims [get_nets CLK_NET*]
1
```

However in the following example, none of the nets named *REG_NET_** are returned to their original state in noise analysis.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets REG_NET_*]
1

pt_shell> remove_si_noise_analysis -aggressors [get_nets REG_NET_*]
Warning: Cannot remove an effect that was not set on net(s) REG_NET_1.
 (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s) REG_NET_2.
 (XTALK-107)
1

pt_shell> remove_si_noise_analysis -aggressors [get_nets REG_NET_*]
-victims [get_nets CLK_NET]
Warning: Cannot remove an effect that was not set on net(s) REG_NET_1
CLK_NET. (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s) REG_NET_2
CLK_NET. (XTALK-107)
1
```

Feedback

If there is a VIC_NET with aggressors AGG_NET_* and you had set pairwise exclusion on the nets with the list of its aggressors, the AGG_NET_* nets cannot be returned to their original state in noise analysis. For example,

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets VIC_NET]
-aggressors [get_attribute -class net [get_nets VIC_NET] aggressors]
1

pt_shell> remove_si_noise_analysis -aggressors [get_nets AGG_NET*]
Warning: Cannot remove an effect that was not set on net(s)
AGG_NET_1. (XTALK-107)
Warning: Cannot remove an effect that was not set on net(s)
AGG_NET_2. (XTALK-107)
1
```

In order to reset the pairwise exclusion on the nets VIC_NET and AGG_NET_*, the pairwise reset should be used as shown below

```
pt_shell> remove_si_noise_analysis -victims [get_nets VIC_NET]
-aggressors [get_nets AGG_NET*]
1
```

To verify if the exclusion was removed on the nets, use the *report_si_noise_analysis* command with the *-excluded* option.

See Also

- read_parasitics
- set_si_noise_analysis
- report_si_noise_analysis
- si_enable_analysis

---

remove_si_noise_disable_statistical

Removes the effect of the *set_si_noise_disable_statistical* command.

Syntax

int *remove_si_noise_disable_statistical*

dnets

Data Types

dnets list

Feedback

Arguments

dnets

A list of nets for which the effect of the *set_si_noise_disable_statistical* command is removed.

Description

Removes the effect of the *set_si_noise_disable_statistical* command.

The *set_si_noise_disable_statistical* command allows you to disable the statistical analysis for the *dnets* option if selected in composite aggressor group for noise analysis.

The *remove_si_noise_disable_statistical* command removes such effect.

Examples

The following example shows how to put net back into statistical analysis when considered as a composite aggressor.

```
pt_shell> remove_si_noise_disable_statistical [get_nets LOGIC1]
1
```

See Also

- set_si_noise_disable_statistical
- report_si_noise_analysis

---

remove_steady_state_resistance

Removes steady state resistance for a library pin or port.

Syntax

int *remove_steady_state_resistance*

```
[-above]
[-below]
[-low]
[-high]
object_list
```

Data Types

object_list	list

Feedback

Arguments

`-above`

Removes the steady state resistance for above ground or power rail noise analysis region.

`-below`

Removes the steady state resistance for below ground or power rail noise analysis region.

`-low`

Removes the steady state resistance for ground rail noise.

`-high`

Removes the steady state resistance for power rail noise. 1.0.

`object_list`

Specifies a list of lib-pins or ports.

Description

This command removes the steady state resistance information set by the *set_steady_state_resistance* command.

Examples

This example removes steady state resistance information for the above the ground rail noise for pin A of library cell IV in the lsi_10k library:

```
pt_shell> remove_steady_state_resistance -above lsi_10k/IV/A
```

See Also

- set_steady_state_resistance

---

remove_target_library_subset

Removes target library subset specifications from the top-level design or hierarchical cells.

Syntax

status *remove_target_library_subset*

```
[-objects objects]
[-top]
```

Data Types

objects	list or collection

Feedback

Arguments

`-objects` *objects*

Removes the subset specifications from the specified hierarchical cells.

If you do not specify this option or the *-top* option, the tool uses the *-top* option by default.

`-top`

Removes the subset specification from the top-level design.

If you do not specify this option or the *-objects* option, the tool uses the *-top* option by default.

Description

This command removes the target library subset specifications from the specified design objects.

This command removes only subsets that were explicitly set on objects with the *set_target_library_subset* command. It does not remove subsets that are implicitly inherited from a parent hierarchical cell.

Examples

The following example removes the target library subset from the top-level hierarchy.

```
prompt> remove_target_library_subset -top
```

The following example removes the target library subset from a hierarchical cell.

```
prompt> remove_target_library_subset -objects [get_cells top/mid]
```

See Also

- set_target_library_subset
- report_target_library_subset

---

remove_upf

Removes the UPF constraints from the design.

Syntax

status *remove_upf*

Description

The *remove_upf* command is used to clean up UPF data for the design.

Feedback

Examples

The following example shows a typical use of the remove_upf command:

```
pt_shell> link
pt_shell> load_upf upf_file_1
pt_shell> remove_upf
pt_shell> load_upf upf_file_2
pt_shell> create_clock -period 1  CLK
```

See Also

- load_upf
- remove_design

---

remove_user_attribute

Removes a user attribute from an object.

Syntax

string *remove_user_attribute*

```
[-quiet]
[-class class_name]
object_spec
attr_name
```

Data Types

class_name	string
object_spec	list
attr_name	string

Arguments

`-quiet`

Does not report any messages.

`-class class_name`

If the *object_spec* option is a name, this is its class. Allowable values are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, or *lib_pin*.

object_spec

Shows objects from which to remove the attribute. Each element in the list is either a collection or a pattern that combines with the *class_name* option to find the objects.

Feedback

attr_name

Provides the name of the attribute.

Description

The *remove_user_attribute* command removes attributes that you set with the *set_user_attribute* command.

You cannot remove application attributes with this command. Each application attribute that can be removed has a command dedicated to it. For example, the *fanout_load* attribute is removed with the *remove_fanout_load* command.

Examples

This example defines an attribute 'X' for cells then sets the value on all cells in this level of the hierarchy. Finally, the example removes the attribute from one cell.

```
pt_shell> define_user_attribute -type int -class cell X
pt_shell> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'
pt_shell> remove_user_attribute [get_cells i1] X
Removed attribute 'X' from 'i1'
```

See Also

- collections
- define_user_attribute
- get_attribute
- list_attributes
- set_user_attribute
- report_attribute

---

remove_waiver

Remove violation waivers satisfying the given requirements.

Syntax

Boolean *remove_waiver*

```
[-names waiver_name_list]
[-rules rule_name_list]
[-design design]
[-scenarios scenario_name_list]
```

Feedback

Data Types

waiver_name_list	list
rule_name_list	list
design	string
scenario_name_list	list

Arguments

`-names` *waiver_name_list*

Remove waivers with names in the given list.

`-rules` *rule_name_list*

Remove waivers of the givens rules.

`-design` *design*

Remove the waivers for the given design only.

`-scenarios` *scenario_name_list*

Remove waivers active in the given scenarios (waivers for scenario-dependent rules only). This option should be used in conjuction of the "-design" option.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Remove waivers with the given names, for the given rules, and active in the given scenarios of the given design. If no *-names* option given, all waiver names are included; if no *-rules* option is given, waivers for all rules are included; if no *-scenarios* option given, waivers active in any scenario, including waivers for scenario-independent rules, are included. If none of the *-names*, *-rules*, *-design*, or *-scenarios* options are given, all existing waivers will be removed.

Note that multiple waivers can be applied that affect same instances. For example, using the *create_waiver -cells* option, instance based waivers can be set on a hierarchical cell and also on a particular instance inside the hierarchical cell. In such cases, all the waivers affecting the instance should be removed before the waiver is completely gone.

Examples

The following example removes a waiver named my_waiver1:

```
pt_shell> remove_waiver -names my_waiver1
```

See Also

- create_waiver
- write_waiver

Feedback

remove_waveform_integrity_analysis

Removes static or dynamic constraints for waveform integrity analysis.

Syntax

string *remove_waveform_integrity_analysis*

```
[-static]
[-dynamic]
[-sms_scenarios sms_scenarios_list]
[object_list]
```

Data Types

object_list	list
sms_scenarios_list	collection

Arguments

`-static`

Removes the constraint for static waveform integrity analysis.

`-dynamic`

Removes the constraint for dynamic waveform integrity analysis.

object_list

Specifies a list of cell input pins.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios. The command uses this option to restrict its scope, removing only the waveform integrity constraints compatible with the specified SMS scenarios. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

This command removes the constraint limit for waveform integrity analysis. The *-static* and *-dynamic* options specify which constraint is removed.

If the list of cell input pins are provided, the constraint limit on the specified cell inputs is removed. If there is no list of cell input pins, the design-level constraint value, if any, is removed.

Examples

The following example removes a static constraint on synchronous input pin "DFF/CLK".

```
pt_shell> remove_waveform_integrity_analysis -static 0.3 [get_pins
 DFF/CLK]
```

Feedback

See Also

- report_constraint
- report_waveform_integrity_analysis
- set_waveform_integrity_analysis

---

remove_wire_load_min_block_size

Removes the minimum block size for automatic wire load selection.

Syntax

int *remove_wire_load_min_block_size*

Arguments

None.

Description

Removes the minimum block area for automatic wire load selection in the current design, set by the *set_wire_load_min_block_size* command.

Examples

The following example removes the previously-set minimum block size.

```
pt_shell> remove_wire_load_min_block_size
```

See Also

- report_wire_load
- set_wire_load_min_block_size
- set_wire_load_selection_group
- auto_wire_load_selection

---

remove_wire_load_model

Removes wire load model from designs, hierarchical cells, or ports.

Syntax

string *remove_wire_load_model*

```
[object_list]
```

Feedback

Data Types

`list object_list`

Arguments

object_list

Lists ports, designs, or hierarchical cells. If this option is not specified, the wire load model is removed from the current instance, or from the current design if current instance is not set.

Description

Removes user-specified wire load model information on designs, ports, or hierarchical cells. The wire load model is used to calculate net capacitance, resistance, and area for designs, which are not placed and routed. You specify wire load models with the *set_wire_load_model* command, or by using automatic wire load selection.

To display wire load model settings for the current design or instance, use *report_wire_load*. To show wire load information for ports, use *report_port -wire_load*.

Examples

The following example removes wire load model settings from the current design and from all hierarchical cells in the design.

```
pt_shell> current_design TOP
pt_shell> remove_wire_load_model
pt_shell> remove_wire_load_model [get_cells -hier * -filter
 "is_hierarchical==true"]
```

The following example removes wire load model settings from port OUT2.

```
pt_shell> remove_wire_load_model [get_ports OUT2]
```

See Also

- report_port
- report_wire_load
- set_wire_load_model
- set_wire_load_selection_group
- report_design
- auto_wire_load_selection

Feedback

remove_wire_load_selection_group

Removes wire load selection_group from current design.

Syntax

status *remove_wire_load_selection_group*

`[object_list]`

Data Types

object_list list

Arguments

object_list

Provides a list of hierarchical cells or designs. If you do not specify the *object_list* option, the wire load selection group is set on the current instance or on the current design if current instance is not set.

Description

This command removes the wire load selection group setting from the current design. Both min and max selection group information is removed. The wire load selection group is specified with *set_wire_load_selection_group*. For more information, see the *set_wire_load_selection_group* man page.

To show the wire load selection group information for a design, use the *report_design* command.

Examples

This example removes the wire load selection group setting from the current design.

```
pt_shell> remove_wire_load_selection_group
```

See Also

- report_design
- set_wire_load_selection_group

rename

Renames or deletes a command.

Syntax

string *rename*

```
old_name
new_name
```

Arguments

old_name

Specifies the current name of the command.

new_name

Specifies the new name of the command.

Description

Renames the *old_name* command so that it is now called *new_name*. If *new_name* is an empty string, then *old_name* is deleted. The *old_name* and *new_name* arguments may include namespace qualifiers (names of containing namespaces). If a command is renamed into a different namespace, future invocations of it will execute in the new namespace. The *rename* command returns an empty string as result.

Note that the *rename* command cannot be used on permanent procedures. Depending on the application, it can be used on all basic builtin commands. In some cases, the application will allow all commands to be renamed.

WARNING: *rename* can have serious consequences if not used correctly. When using *rename* on anything other than a user-defined Tcl procedure, you will be warned. The *rename* command is intended as a means to wrap other commands: that is, the command is replaced by a Tcl procedure which calls the original. Parts of the application are written as Tcl procedures, and these procedures can use any command. Commands like *puts*, *echo*, *open*, *close*, *source* and many others are often used within the application. Use *rename* with extreme care and at your own risk. Consider using *alias*, Tcl procedures, or a private namespace before using *rename*.

Examples

This example renames my_proc to my_proc2:

```
prompt> proc my_proc {} {echo "Hello"}
prompt> rename my_proc my_proc2
prompt> my_proc2
Hello
prompt> my_proc
Error: unknown command 'my_proc' (CMD-005).
```

See Also

- define_proc_attributes

Feedback

rename_cell

Changes the name of a cell.

Syntax

int *rename_cell*

```
cell
new_name
```

Data Types

cell	list
new_name	string

Arguments

cell

Specifies one cell to be renamed.

new_name

Specifies the new name for *cell*.

Description

The *rename_cell* command changes the name of a cell. Either a cell name that matches a single cell, or a collection of one cell, is passed in as the *cell* argument.

The *rename_cell* command fails if any of the following are true:

- PrimeTime cannot find *cell*.
- PrimeTime finds multiple cells that match *cell*.
- PrimeTime finds a leaf cell matching *new_name*.
- *new_name* is not at the same level of hierarchy as *cell*.

The *write_changes* command records the *rename_cell* command.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Feedback

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Examples

In the following examples, a cell is renamed in the current instance.

```
pt_shell> rename_cell cellA cellB
Information: Renamed cell 'cellA' to 'cellB' in design 'top'. (NED-008)
1
```

In the following examples, a cell is renamed at a level below the current instance. Currently, H1 and H1/H2 are instances of designs that have multiple instances. Therefore, they are uniquified as part of the editing operation.

```
pt_shell> rename_cell H1/H2/u1 H1/H2/cellC
Uniquifying 'H1/H2' (low) as 'low_0'.
Uniquifying 'H1' (inter) as 'inter_0'.
```

Feedback

```
Information: Renamed cell 'u1 to 'cellC' in 'top/H1/H2'. (NED-008)
1
```

See Also

- rename_design
- rename_net
- write_changes

---

rename_design

Change the name of a design.

Syntax

int *rename_design*

```
design
new_name
```

Data Types

design	list
new_name	string

Arguments

design

Specifies a design to be renamed. Only one design can be specified.

new_name

Specifies the new name for the *design* option.

Description

The *rename_design* command changes the name of a design. Either a design name which matches a single design, or a collection of one design, is passed in as the *design* argument. The command allows only simple design name changes for a single design. You cannot change the association of a design with its source file.

For the *design* to be renamed to *new_name*, there cannot be a design named *new_name* in the same file as *design* (see the Examples). The command may also fail if there are any instances of *design* in a linked design.

Renaming designs in PrimeTime should be done prior to any linking to avoid the case of instantiation conflicts.

Feedback

Note that the *rename_design* command is not recorded for output with the *write_changes* command.

Examples

In the following example, design *top* is successfully renamed to *top2*:

```
pt_shell> list_designs
Design Registry:
 *  top            /home/designs/top.v:top
1
pt_shell> rename_design [get_designs top] top2
Renamed design 'top' to 'top2'
1
pt_shell> list_designs
Design Registry:
 *  top2           /home/designs/top.v:top2
1
```

In the following example, design *fbox* cannot be renamed *mbox* because there is already a design of that name in the same file. An attempt to rename it to *fbox2* fails because *fbox* is instantiated in a linked design.

```
pt_shell> list_designs
Design Registry:
    fbox           /home/designs/top.v:fbox
    mbox           /home/designs/top.v:mbox
 *L top            /home/designs/top.v:top
1
pt_shell> rename_design fbox mbox
Error: Could not rename design 'fbox' to 'mbox':
 design 'mbox' already exists in file
        '/home/ajs/designs/top.v' (NED-020)
0
pt_shell> rename_design fbox fbox2
Error: Could not rename design 'fbox' to 'fbox2':
 design 'fbox' is instantiated in design 'top' (NED-020)
0
```

See Also

- list_designs
- link_design
- write_changes

---

rename_net

Changes the name of a net.

Feedback

Syntax

status *rename_net*

```
net
new_name
```

Data Types

net	list
new_name	string

Arguments

net

Specifies a net to be renamed. Only one net can be specified.

new_name

Specifies the new name for *net*.

Description

The *rename_net* command changes the name of a net. Either a net name that matches a single net or a collection of one net is passed in as the *net* argument.

The *rename_net* command fails if any of the following are true:

- PrimeTime cannot find *net*.
- PrimeTime finds multiple nets that match *net*.
- PrimeTime finds a net, port, or hierarchical pin matching *new_name* in the same hierarchical block as *net*.
- *new_name* is not at the same level of hierarchy as *net*.

The *rename_net* command is recorded for output with the *write_changes* command.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is re-linked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Examples

In the following examples, a net is renamed in the current instance.

```
pt_shell> rename_net netA netB
Information: Renamed net netA' to 'netB' in design 'top'. (NED-009)
1
```

In the following examples, a net is renamed at a level below the current instance. Currently, H1 and H1/H2 are instances of designs that have multiple instances. Therefore, they are uniquified as part of the editing operation.

```
pt_shell> rename_net H1/H2/w1 H1/H2/netC
Uniquifying H1/H2 (low) as 'low_0'.
Uniquifying 'H1' (inter) as 'inter_0'.
Information: Renamed net 'w1' to 'netC' in 'top/H1/H2'. (NED-009)
1
```

Feedback

See Also

- rename_cell
- rename_design
- write_changes

---

report_activity_annotation

Reports the activity conflict among different segments of a net in VCD or FSDB file.

Syntax

```
int report_activity_annotation
[-debug_file]
[-verbose]
```

Data Types

net_list	list

Arguments

`-debug_file`

To specify the name of the output file which will be dumped out by the command.

`-verbose`

To print out all net/pin activities including non-conflict activities on the net.

Description

To check and report conflicting activities among different segments of a net in any timestamp in a VCD/FSDB file. You need to read VCD or FSDB file first, and then run the command. There is no need to run update_power.

This command should only be run in time based mode.

The conflict values in each timestamp among different net segments and pins are reported in an output file. The default file name is "activity_annotation_report.rpt".

The *debug_file* option can be used to change the output report file name. The *net_list* option can be used to report specific nets in the design.

Examples

The following example reports conflict on all nets in the design. The command dumps out the conflict output to a default file name "activity_annotation_report.rpt".

Feedback

```
pwr_shell> report_activity_annotation  [get_nets *]
report_activity_annotation [get_nets *]

Information : Activity annotation report 'activity_annotation_report.rpt'
 generated successfully

1
```

The following example reports the activity conflict for net "s1" and creates an output file "debug_s1_report.rpt"

```
pwr_shell> report_activity_annotation [get_net s1]

report_activity_annotation -debug_file debug_s1 [get_net s1]
Information : Activity annotation report 'debug_s1_report.rpt' generated
 successfully
1
```

See Also

- get_switching_activity
- check_activity

---

report_activity_derate

Reports the clock activity derating factors for a specified list of instances.

Syntax

```
int report_activity_derate
[-cell_list]
```

Data Types

cell_list	list

Arguments

-cell_list

Reports the *derate_value* currently applied to cell instances specified in the cell list.

Description

Reports the toggle rate derating factors currently set on a cell instance or a list of cells in the current design.

Feedback

Clock activity derating factor affects toggle rate on cells in the clock network. The clock derating factors are used as a scalar multiplicative factor in calculation of toggle rate for output net of cell. If clock derating factors are not specified, the value of 1.0 is assumed.

The *cell_list* option can be used to report specific derating factors on instances (cells) in the design.

Examples

The following example reports clock activity toggle rate derating factor for all cells in the design.

```
pwr_shell> report_activity_derate  [get_cells *]
report_activity_derate [get_cells *]
****************************************
Report : Clock activity derate factors
Version: O-2018.06-ALPHA-20180405
****************************************
Cell     Derate factor
--------------------------------------------------------------------------------

i_ff_2     0.000000
i_ff_3     0.000000
i_ff_4     0.000000
i_ff_1     0.000000
i_or_1     0.100000
i_icg_1     0.200000
i_and_1     0.100000
--------------------------------------------------------------------------------
1
```

The following example reports the toggle rate derate factor set for integrated clock gating cell "i_icg_1".

```
pwr_shell> report_activity_derate [get_cell i_icg_1]
report_activity_derate [get_cell i_icg_1]
****************************************
Report : Clock activity derate factors
****************************************
Cell     Derate factor
--------------------------------------------------------------------------------

i_icg_1     0.200000
--------------------------------------------------------------------------------
1
```

See Also

- set_switching_activity
- reset_switching_activity

Feedback

report_activity_file_check

Reads an activity file without annotating the switching activity, and reports the proceedings of the mapping.

Syntax

string *report_activity_file_check*

```
[-rtl]
[-format SAIF | VCD | FSDB]
[-pipe_exec pipe]
[-path prefix]
-strip_path strip_path
[-nosplit]
[-find pattern]
file_name
```

Data Types

pipe	string
prefix	string
strip_path	string
pattern	string
file_name	string

Arguments

`-rtl`

Specifies that the name mapping method must be applied during the *time_based* power analysis mode. As the name mapping method is always applied in the averaged power analysis mode, the -rtl option is disabled in the averaged power analysis mode. For more information about specifying name mapping, see the *set_rtl_to_gate_name* man page. *set_rtl_to_gate_name* man page.

`-format SAIF | VCD | FSDB`

Specifies whether the file referenced by the *file_name* argument is a FSDB file or a VCD file or a SAIF file. The *format* argument is not necessary if the format of the *file_name* can be recognized by the file name extension.

`-pipe_exec pipe`

Specifies a shell command used to generate the VCD file using the *file_name* option. This option invokes the shell command and directly pipes the output VCD file to PrimePower. There is no VCD disk file generated at all. You cannot use this option if the value specified with the *format* argument is SAIF.

`-path` *`prefix`*

Specifies a relative path from the current design to the hierarchical low- level design for which the VCD file has been created. By default, the absolute path names are used. Use this option if the VCD file refers to an object in a hierarchy. Do not use this option if the VCD file refers to an absolute path.

`-strip_path` *`strip_path`*

Specifies a path prefix that must be stripped from all the object names read from the VCD or SAIF file. This option is applied to strip the testbench or instance path from the RTL VCD or SAIF file.

`-nosplit`

Prevents line splitting if a column overflows.

`-find` *`pattern`*

Reports only on the activity file objects that match the *pattern* option, so the report can be much shorter and more readable. The pattern is a glob; for example, the pattern "A/B*" matches "A/B1", "A/B2", and "A/B/C/D". Multiple globs can be included with commas, such as {A/B*, C/*/Q}. Only the activity file objects are checked to see if the pattern matches. Any activity file object not matching the pattern is not reported.

`file_name`

Specifies the name of the switching activity file to be read. The specified file can be a SAIF, VCD, VPD, or FSDB file and can be of compressed format. The file name extension can be used to determine the file type. The file names with the ".vcd", ".vcd.gz", ".vcd.Z", ".vcd.bz2", ".vpd" or ".fsdb" extensions are converted to RTL VCD files and read as RTL VCD files; whereas the file names with the ".saif", ".saif.Z", ".saif.gz", or ".saif.bz2" extensions are converted to RTL SAIF files and read as RTL SAIF files.

Description

The *report_activity_file_check* command helps debug the switching activity annotation from simulation. The command processes the activity file in the same way as the *read_saif* or *read_vcd* command would, except that annotation is not applied to the design. This command reports the proceedings of the name mapping. For each hierarchical module (in VCD) or instance (in SAIF), the corresponding hierarchical cell in the design is displayed, if found. For each signal (in VCD) or net (in SAIF), the corresponding net or pin in the design is shown, if any. The exact method (or the combination of methods) used to arrive at the mapping is also reported. In addition, this command also reports information about any logical inversion between the signal in VCD (or net in SAIF) and the net or pin in the design.

Feedback

Using the *report_activity_file_check* command, it is possible to determine where in the design a particular signal in a VCD would be annotated by the *read_vcd* command. To do this, run the *report_activity_file_check* command with the -find option to output only the results matching the signal in question.

When you set the *power_analysis_mode* variable to *time_based*, you cannot use the SAIF files with the *report_activity_file_check* command.

Examples

The following example reports the mapping of a SAIF file to the design. Note that nothing is reported for the signal Ud SAIF file.

```
pt_shell> report_activity_file_check my_file.saif -strip_path tb/dut
 -path top
****************************************
Report : Activity File Matching Check
 my_file.saif
 -strip_path tb/dut
 -path top
Design : top
Version: F-2011.06-Beta
Date   : Tue Apr 12 17:23:18 2011
****************************************
--------------------------------------------------------------------------------
---------------

      Name Mapping Methods
      --------------------
       d - Default name mapping
       e - Exact name mapping
       m - User-defined rtl to gate name mapping
       s - User-defined mapping by string substitution

      Attributes
      ----------
       v - Inverted
--------------------------------------------------------------------------------
------------------
File Type: saif
--------------------------------------------------------------------------------
-------------------
Activity    Activity                        Design     Design
 Name      Attributes
File         File                           Object     Object
 Mapping
Object Type Name                            Type       Name
 Method
--------------------------------------------------------------------------------
--------------------
instance    (strip_path matches)            Design
net         Ua                              Net        Ua/Q            e
```

```
net         Ub                                  Net         Ub/Q            e
net         Ud                                  Net         None Found
net         clk                                 Net         clk             e
net         in                                  Net         in              e
--------------------------------------------------------------------------------
-----------------
```

The following example reports the mapping of a SAIF file to the design after adding a name-mapping command to fix the missing switching activity annotation to the signal Ud.

```
pt_shell> set_rtl_to_gate_name -rtl {Ud} -gate {Uc/Q}
1
pt_shell> report_activity_file_check my_file.saif -strip_path tb/dut
 -path top


****************************************
Report : Activity File Matching Check
 my_file.saif
 -strip_path tb/dut
 -path top
Design : top
Version: F-2011.06-Beta
Date   : Tue Apr 12 17:23:18 2011
****************************************
--------------------------------------------------------------------------------
---------------
      Name Mapping Methods
      --------------------
       d - Default name mapping
       e - Exact name mapping
       m - User-defined rtl to gate name mapping
       s - User-defined mapping by string substitution

      Attributes
      ----------
       v - Inverted
--------------------------------------------------------------------------------
---------------
File Type: saif
--------------------------------------------------------------------------------
---------------
Activity    Activity                        Design     Design        Name
     Attributes
File        File                            Object     Object
 Mapping
Object Type Name                            Type       Name
 Method
--------------------------------------------------------------------------------
---------------
instance    (strip_path matches)            Design
net         Ua                              Net        Ua/Q         e
net         Ub                              Net        Ub/Q         e
```

```
net         Ud                                Net         Uc/Q    m
net         clk                               Net         clk          e
net         in                                Net         in           e
-------------------------------------------------------------------------
-----------------
```

The following example reports the mapping of a SAIF file to the design, and limits the output by using the -find option.

```
pt_shell> report_activity_file_check my_file.saif -strip_path tb/dut
 -path top -find U*

****************************************
Report : Activity File Matching Check
 my_file.saif
 -strip_path tb/dut
 -path top
Design : top
Version: F-2011.06-Beta
Date   : Tue Apr 12 17:23:18 2011
****************************************
-------------------------------------------------------------------------

      Name Mapping Methods
      --------------------
       d - Default name mapping
       e - Exact name mapping
       m - User-defined rtl to gate name mapping
       s - User-defined mapping by string substitution

      Attributes
      ----------
       v - Inverted
-------------------------------------------------------------------------
File Type: saif
-------------------------------------------------------------------------
Activity    Activity   Design     Design      Name      Attributes
File        File       Object     Object      Mapping
Object Type Name       Type       Name        Method
-------------------------------------------------------------------------
net         Ua         Net        Ua/Q        e
net         Ub         Net        Ub/Q        e
net         Ud         Net        Uc/Q        m
-------------------------------------------------------------------------
```

The following example reports the mapping a FSDB file to the design.

```
pt_shell> report_activity_file_check my_file.fsdb -strip_path tb/dut
 -path top

****************************************
Report : Activity File Matching Check
  my_file.fsdb
```

Feedback

```
  -strip_path tb/dut
Design : top
Version: ...
Date   : ...
*************************************
------------------------------------------------------------------------

      Name Mapping Methods
      --------------------
       d - Default name mapping
       e - Exact name mapping
       m - User-defined rtl to gate name mapping
       s - User-defined mapping by string substitution

      Attributes
      ----------
       v - Inverted
------------------------------------------------------------------------
File Type: fsdb
------------------------------------------------------------------------
Activity    Activity   Design     Design      Name      Attributes
File        File       Object     Object      Mapping
Object Type Name       Type       Name        Method
------------------------------------------------------------------------
net         Ua         Net        Ua/Q        e
net         Ub         Net        Ub/Q        e
net         Ud         Net        Uc/Q        m
------------------------------------------------------------------------
```

See Also

- set_rtl_to_gate_name
- read_vcd
- read_saif
- report_name_mapping
- power_analysis_mode

---

report_activity_waveforms

Reports on activity analysis of VCD.

Syntax

int *report_activity_waveforms*

```
[-nosplit]
```

Feedback

Arguments

`-nosplit`

Specifies that lines with overflow should not be split. This can be useful when the output is read by a script.

Description

After running the *write_activity_waveforms* command, you can use the *report_activity_waveforms* command to summarize the results and give the peak activity times for each block in the VCD file analyzed.

Examples

This example shows the form of the report.

```
pt_shell> write_activity_waveforms -output filename1 -vcd my_vcd.vcd
 -interval 10
pt_shell> report_activity_waveforms

****************************************
Report : Time-Based Switching Activity
Activity File: my_vcd.vcd
Version: B-2008.06-Dev-DEV
Date   : Wed Apr  9 17:54:06 2008
****************************************

Block                              # Signals      Peak           Peak
      Peak
Name                                            Interval       Interval
      Toggle
                                                Start          End
      Rate
--------------------------------------------------------------------------
------------------
TOP_level_of_my_design             1383           0              2000
      0.000192
 Second_level_of_my_design         1335           0              2000
      0.000199
--------------------------------------------------------------------------
------------------
```

See Also

- write_activity_waveforms

---

report_alternative_lib_cells

Generates a report on the timing effects of alternative library cells that can replace an existing cell instance.

Feedback

Syntax

string *report_alternative_lib_cells*

```
[-current_library]
[-libraries lib_spec]
[-delay_type delay_type]
[-all_pins]
[-weighted_design_cost]
[-total_design_cost]
[-significant_digits digits]
[-nosplit]
cell
```

Data Types

lib_spec	list
delay_type	string
digits	integer
cell	string

Arguments

`-current_library`

Restricts the report to new library cells only from the same library as that of the cell being considered for replacement.

`-libraries lib_spec`

Specifies a list of libraries from which to select alternative library cells. You can specify either a list of library names or a collection created by the *get_libs* command. The default is to select from all libraries specified in the *link_path* variable.

`-delay_type delay_type`

Specifies the type timing slack to report: *max* (the default), *min*, *min_max*, *max_rise*, *max_fall*, *min_rise*, or *min_fall*. For the *min* and *max* setting, the worse between rising and falling slacks is reported. For the *min_max* setting, both min and max slacks are reported.

`-all_pins`

Reports the worst slack through each pin of the cell. By default, the command reports only the worst slack of the single worst output pin.

`-weighted_design_cost`

Reports the weighted design cost (negative slack) caused by using each alternative library cell. The value is calculated as a weighted sum over the weighted cost values calculated for all path groups. The reported cost values are the same as those reported by the *report_constraint* command for minimum

delay (costs resulting from hold constraints) and maximum delay (costs resulting from setup constraints).

The minimum delay cost is calculated as follows:

```
weighted min_delay/hold cost = summation over all path groups
( group weight * sum of all nonzero hold costs in group )
```

The maximum delay cost is calculated as follows:

```
weighted max_delay/setup cost = summation over all path groups
( group weight * worst setup cost in group )
```

Maximum cost values are reported by default or when *-delay_type max*, *max_rise*, or *max_fall* is specified.

Minimum cost values are reported when *-delay_type min*, *min_rise*, or *min_fall* is specified.

If *-delay_type min_max* is specified, both minimum and maximum costs are reported.

`-total_design_cost`

Reports the total design cost (negative slack) caused by using each alternative library cell. This option is similar to the *-weighted_design_cost* option, except that the total design cost can change in cases when the weighted maximum delay (setup) cost does not. The total cost can give some indication of the overall improvement in the design timing when the change does not improve the worst setup timing violation.

The total design cost is also computed as a weighted sum of the total costs computed for each path group. Only positive cost values are included in the calculation. It is calculated as follows:

```
total design cost = summation over all path groups
( group weight * group total cost )

group total cost = summation over all group endpoints
having positive cost ( cost value )
```

`-significant_digits` *`digits`*

Specifies the number of digits to the right of the decimal point that are reported for slack values. Allowed values are 0-13. The default is determined by the *report_default_significant_digits* variable, default 2.

`-nosplit`

Prevents splitting of long lines in the report.

cell

Specifies a single cell instance in the design. The command reports the timing effects of replacing that cell with alternative cells.

Description

The *report_alternative_lib_cells* command generates a report to help you select an alternative library cell for a given cell instance. The report shows the slack values that would occur by replacing the cell with alternative cells using the *size_cell* command. The command performs the replacement temporarily using each alternative cell and reports the resulting worst timing slack.

By default, the command reports the worst rising or falling max (setup) slack to the single output pin with the worst slack. To report a different type of slack, use the *-delay_type* option. To report the worst slack through each pin of the cell, use the *-all_pins* option.

The command uses the same criteria to find functionally equivalent alternative cells as the *size_cell* and *get_alternative_lib_cells* commands, and searches through the available libraries in the same order. For details, see the *size_cell* man page.

Examples

The following example reports the alternative library cells for cell instance U141 and shows the min and max slacks resulting from each alternative, including the current cell, which is marked with an asterisk.

```
pt_shell> report_alternative_lib_cells -delay_type min_max U141
Information: Invalidating logical update. (PTE-139)
Information: Updating design - Started                (UITE-214)
Information: Updating design -   Calculating delays   (UITE-214)
 ...
Information: Updating design - Completed              (UITE-214)
 ...
*************************************

Alternative        Slack (min:max)
Library Cells
-------------------------------------------------------------------
cb13_max/or02d4    3.752(r):4.178(r)
cb13_max/or02d7    3.150(r):5.101(r)
cb13_max/or02da *  2.935(r):5.386(r)
cb13_max/or02d2    5.006(r):1.596(r)
cb13_max/or02d1    7.674(r):-3.834(r)
cb13_max/or02d0    13.465(r):-15.573(r)

*: The original library cell, not an alternative.
```

The following example generates a report that shows the weighted and total design costs.

```
pt_shell> set libs_in_mem [get_libs *]
```

Feedback

```
pt_shell> report_alternative_lib_cells U155 -libraries $libs_in_mem
              -delay_type max -weighted_design_cost -total_design_cost
 ...
**************************************

Alternative          Slack       Weighted           Total Design Cost
Library Cells                    max_delay/setup
                                 Cost
-------------------------------------------------------------------------
dz13_max/and2_2      1.01(r)     4.03               35.67
dz13_max/and2_1     -1.34(f)     4.03               34.34
dz13_max/and2_0 *   -2.34(r)     4.03               36.78
dz13_max/and2_3     -3.01(f)     4.03               37.88

*: The original library cell, not an alternative.
```

The weighted design cost is not affected by any of the alternatives, but the total cost of the design can be improved by using library cell and2_2 or and2_1 in place of the current library cell and2_0.

See Also

- get_alternative_lib_cells
- get_libs
- set_min_library
- size_cell
- link_path

---

report_analysis_coverage

Generates a report about the coverage of timing checks.

Syntax

status *report_analysis_coverage*

```
[-status_details status_list]
[-check_type check_type_list]
[-exclude_untested untested_reason_list]
[-sort_by sort_method]
[-significant_digits digits]
[-nosplit]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-sms_scenarios sms_scenarios_list]
```

Feedback

Data Types

status_list	list
check_type_list	list
untested_reason_list	list
sort_method	string
digits	integer
pre_command_string	string
post_command_string	string
sms_scenarios_list	collection

Arguments

`-status_details` *status_list*

Specifies a space-delimited list of status names about which to show detail in the report. The allowed values are *untested*, *violated*, and *met*. By default, only summary information is shown. Use this option to see information about individual timing checks.

`-check_type` *check_type_list*

Specifies a list of check types to include in the report. The allowed values are *setup*, *hold*, *recovery*, *removal*, *min_period*, *min_pulse_width*, *clock_separation*, *data_separation*, *max_skew*, *clock_gating_setup*, *clock_gating_hold*, *out_setup*, *out_hold*, and *nochange*.

Check types *out_setup* and *out_hold* refer to the output setup and output hold constraints generated by the *set_output_delay* command.

`-exclude_untested` *untested_reason_list*

Specifies a space-delimited list of reasons to exclude an untested check in the report. A check classified as untested is excluded from the report for any of the reasons specified as values to this argument. Therefore, the coverage statistics are unaffected by these excluded constraints. The following values are allowed:

- *constant_disabled* - Paths to this check are disabled because of case analysis or a logic constant propagated through the design (for example, caused by a signal tied high or low).
- *mode_disabled* - A timing constraint is disabled because it requires a mode to be selected in mode analysis, and that mode is not selected.
- *user_disabled* - The timing check is explicitly disabled by the user.
- *no_paths* - The timing check had no paths found to it and as a result there were no arrival times.
- *false_paths* - All paths were false to a constrained pin.

- *no_endpoint_clock* - The timing check has no destination clock signal to latch the data.
- *no_startpoint_clock* - The timing check has no clock that launches the data at a startpoint latch.
- *no_constrained_clock* - There is no constrained clock for skew or clock separation checks.
- *no_ref_clock* - There is no reference clock for skew or clock separation checks.
- *no_clock* - A minimum pulse width or period width check has no clock.
- *no_min_check* - An external delay was specified with a maximum delay but no minimum delay.
- *no_max_check* - An external delay was specified with a minimum delay but no maximum delay.
- *unknown* - The reason is not listed above.

`-sort_by` *`sort_method`*

Specifies the sorting method for the output of the detailed list. The following values are allowed:

- *slack* (the default) - Sorts by the slack (untested is treated as negative infinity slack), followed by the pin name and check type.
- *name* - Sorts alphabetically by the name of the constrained pin, followed by the slack and check type.
- *check_type* or *check* - Sorts alphabetically by the check type, followed by the slack and pin name.

`-significant_digits` *`digits`*

Specifies the number of digits to the right of the decimal point that are to be reported. The allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default is 2. Use this option if you want to override the default.

`-nosplit`

Prevents line-splitting when columns overflow.

`-pre_commands` *`pre_command_string`*

This option is available only if you invoke PrimeTime with the *-multi_scenario* option. This option allows you to specify a string of commands to be executed in the worker context before the execution of the merged reporting command.

Feedback

Commands must be grouped using the ";" character. The maximum size of a command is 1000 characters.

`-post_commands` *`post_command_string`*

This option is available only if you invoke PrimeTime with the *-multi_scenario* option. This option allows you to specify a string of commands to be executed in the worker context after the execution of the merged reporting command. Commands are grouped using the ";" character. The maximum size of a command is 1000 characters.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only timing checks compatible to the specified SMS scenarios collection are reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

Generates a report showing information about the coverage of timing checks in the current design, or current instance if it is defined. The default report is a summary of checks by type. For each type of check (for example, *setup*), the report shows the number and percentage of checks meeting and violating constraints, and those that are untested; the report does not show check types for which there are no checks. The report does not show unconstrained output ports; that is, those that have no output delay or max_delay or min_delay; however, the report does show constrained output ports.

For more information about the individual timing checks, use the *-status_details* option, which shows all checks with the corresponding status. Untested checks show information about the reason why they are untested, if the reason can be determined.

By default, the coverage statistics include constraints that have been disabled. Constraints could be disabled because of constant propagation (for logic constants in the design and case analysis), modes that are not enabled, or explicit disabling of timing arcs by the user. Such disabled constraints appear as untested in the coverage statistics. If you want to exclude some of these untested constraints from the coverage statistics, use the *-exclude_untested* option.

Note that the command reports primarily library defined timing checks, in addition to *output_setup* and *output_hold*, which can be set by the *set_output_delay* command. User-defined constraints such as those set by *set_max_delay*/*set_min_delay* commands are not reported by *report_analysis_coverage*.

Examples

The following example shows the summary report.

```
pt_shell> report_analysis_coverage
****************************************
```

```
Report : analysis_coverage
Design : counter
 ...
****************************************

Type of Check         Total      Met          Violated    Untested
-------------------------------------------------------------------
setup                  5         0 (  0%)     3 ( 60%)    2 ( 40%)
hold                   5         3 ( 60%)     0 (  0%)    2 ( 40%)
-------------------------------------------------------------------
All Checks            10         3 ( 30%)     3 ( 30%)    4 ( 40%)
```

The following example shows the detailed report of untested setup checks.

```
pt_shell> report_analysis_coverage -status_details {untested} -check_type
 {setup}
****************************************
Report : analysis_coverage
        -status_details {untested }
        -sort_by slack
        -check_type {setup }
Design : counter
 ...
****************************************

Type of Check         Total              Met         Violated
 Untested
-------------------------------------------------------------------------------
-------
setup                     5         0 (  0%)         3 ( 60%)         2
 ( 40%)
-------------------------------------------------------------------------------
-------
All Checks                5         0 (  0%)         3 ( 60%)         2
 ( 40%)


Constrained                            Related   Check
Pin                                    Pin       Type         Slack
 Reason
-------------------------------------------------------------------------------
-------
ffd/CR                                 CP        setup     untested
 no_clock
ffd/D                                  CP        setup     untested
 no_clock
```

The following example shows generation of a report that includes details of untested and violated checks, excludes timing checks if they are untested because of constant propagation, sorts by slack, and shows the *setup* and *out_setup* checks.

Feedback

```
pt_shell> report_analysis_coverage -check_type {out_setup setup} \\
          -exclude_untested {constant_disabled} -status_details {violated
 untested}
****************************************
Report : analysis_coverage
        -status_details {untested violated }
        -exclude_untested {constant_disabled}
        -sort_by slack
        -check_type {setup out_setup}
Design : seq_case
 ...
****************************************

Type of Check         Total              Met         Violated
 Untested
------------------------------------------------------------------------
-------
setup                     6         5 ( 83%)         1 ( 17%)         0 (
  0%)
out_setup                 5         4 ( 80%)         0 (  0%)         1
 ( 20%)
------------------------------------------------------------------------
-------
All Checks               11         9 ( 82%)         1 (  9%)         1 (
  9%)


Constrained                            Related   Check
Pin                                    Pin       Type         Slack
 Reason
------------------------------------------------------------------------
-------
Q2                                               out_setup untested
 no_paths
ff4/TI                                 CP        setup        -2.30
```

See Also

- check_timing
- current_instance
- report_constraint
- report_timing
- report_default_significant_digits

---

report_annotated_check

Reports back-annotated timing checks.

Feedback

Syntax

string *report_annotated_check*

```
[-setup]
[-hold]
[-recovery]
[-removal]
[-nochange]
[-width]
[-period]
[-max_skew]
[-clock_separation]
[-max_line num]
[-list_annotated]
[-list_not_annotated]
[-list_lib_annotated]
[-constant_arcs]
[-significant_digits digits]
```

Data Types

num int

Arguments

`-setup`

Reports setup timing checks.

`-hold`

Reports hold timing checks.

`-recovery`

Reports recovery timing checks.

`-removal`

Reports removal timing checks.

`-nochange`

Reports nochange timing checks.

`-width`

Reports minimum pulse width timing checks.

`-period`

Reports minimum period timing checks.

`-max_skew`

Reports maximum skew timing checks.

`-clock_separation`

Reports clock separation timing checks.

`-max_line` *num*

Provides maximum number of line for the *-list_** options.

`-list_annotated`

Lists timing arcs that are back-annotated. Use this option if no annotated checks are expected to identify the specified annotated checks.

`-list_not_annotated`

Lists timing arcs that are not back-annotated. Use this option if some annotated checks are missing to identify the missing annotated checks.

`-list_lib_annotated`

Lists library timing arcs that are back-annotated.

`-constant_arcs`

Keeps a separate count for the arcs that are disabled due to logic constants (but not case_analysis). With this option, the *-list_annotated* or *-list_not_annotated* option does not list the arc disabled due to logic constants. The total count at the end of the table includes the constant_arcs count.

`-significant_digits` *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13. If you do not use this option, the number of digits is specified by the *report_default_significant_digits* variable, which defaults to 2. Use this option if you want to override the default.

Description

Provides a summary report of how many cell timing checks are annotated in the current design. The purpose of this command is to check if all timing checks of the design are annotated after reading an SDF file. You can use the *-list_not_annotated* option to identify which check timing arcs are not annotated.

Examples

The following are examples of annotated reports.

```
pt_shell> report_annotated_check

****************************
report: annotated_check
****************************

                                |            |            |     NOT     |
```

```
                              |   Total    | Annotated | Annotated |
------------------------------+------------+-----------+-----------+
cell setup arcs               |          2 |         0 |         2 |
cell hold arcs                |          2 |         0 |         2 |
cell recovery arcs            |          2 |         0 |         2 |
cell removal arcs             |          2 |         0 |         2 |
cell nochange arcs            |          2 |         0 |         2 |
cell min pulse width arcs     |          2 |         0 |         2 |
cell min period arcs          |          2 |         0 |         2 |
cell max skew arcs            |          2 |         0 |         2 |
cell clock separation arcs    |          2 |         0 |         2 |
------------------------------+------------+-----------+-----------+
                              |         18 |         0 |        18 |
```

```
pt_shell> report_annotated_check -constant_arcs

****************************
report: annotated_check
****************************

                              |            |           |    NOT    |
                              |   Total    | Annotated | Annotated |
------------------------------+------------+-----------+-----------+
cell setup arcs               |          2 |         0 |         1 |
constant arcs                 |            |         0 |         1 |
cell hold arcs                |          2 |         0 |         1 |
constant arcs                 |            |         0 |         1 |
cell recovery arcs            |          2 |         0 |         2 |
constant arcs                 |            |         0 |         0 |
cell removal arcs             |          2 |         0 |         2 |
constant arcs                 |            |         0 |         0 |
cell nochange arcs            |          2 |         0 |         2 |
constant arcs                 |            |         0 |         0 |
cell min pulse width arcs     |          2 |         0 |         2 |
constant arcs                 |            |         0 |         0 |
cell min period arcs          |          2 |         0 |         2 |
constant arcs                 |            |         0 |         0 |
cell max skew arcs            |          2 |         0 |         2 |
constant arcs                 |            |         0 |         0 |
cell clock separation arcs    |          2 |         0 |         2 |
constant arcs                 |            |         0 |         0 |
------------------------------+------------+-----------+-----------+
                              |         18 |         0 |        18 |
```

See Also

- read_sdf
- set_annotated_check

Feedback

- remove_annotated_check
- reset_design

---

report_annotated_delay

Reports back-annotated delays.

Syntax

string *report_annotated_delay*

```
[-cell]
[-net]
[-from_in_ports]
[-to_out_ports]
[-max_line num]
[-list_annotated]
[-list_not_annotated]
[-constant_arcs]
```

Data Types

```
num     integer
```

Arguments

`-cell`

Reports all data annotated on cells.

`-net`

Reports all data annotated on nets.

`-from_in_ports`

Includes the nets that start from input ports. By default, nets that start at input ports are not included in the list of nets reported for annotated delays. The reason is that SDF format specifies the nets starting at input ports of the design are not part of the SDF file because the delay of this net is dependent on the external environment.

`-to_out_ports`

Includes the nets that end at output ports. By default, nets that end at output ports are not included in the list of nets reported for annotated delays. The reason is that SDF format specifies the nets ending at output ports of the design are not part of the SDF file because the delay of this net is dependent on the external environment.

`-max_line` *num*

Reports the specified maximum number of lines for the *-list_annotated* and *-list_not_annotated* options.

`-list_annotated`

Lists timing arcs that are back-annotated. Use this option to identify the specified annotated delays if no annotated delays are expected. This option lists both connected and unconnected cells.

`-list_not_annotated`

Lists timing arcs that are not back-annotated. Use this option to identify which delay arcs are not annotated. This option lists both connected and unconnected cells.

`-constant_arcs`

Keeps a separate count for the arcs that are disabled due to logic constants (but not case_analysis). With this option, the *-list_annotated* or *-list_not_annotated* option does not list the arc disabled due to logic constants. The total count at the end of the table includes the number of constant arcs.

Description

Provides a summary report of how many cell and net delays are annotated in the current design. The purpose of this command is to check if all delays of the design are annotated after reading an SDF file. By default, net delay starting from input ports or ending at output ports are considered separately.

By default, this command reports annotated data on both cells and nets. To report annotated data on cells or nets only, use the *-cell* or *-net* option, respectively.

Examples

The following is an example of a default report showing annotated delays:

```
pt_shell> report_annotated_delay

****************************
report: annotated_delay
****************************

                             |           |           |    NOT    |
                             |   Total   | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
cell arcs                    |        19 |        19 |         0 |
cell arcs (unconnected)      |         1 |         1 |         0 |
internal net arcs            |         3 |         3 |         0 |
net arcs from primary inputs |        11 |        11 |         0 |
net arcs to primary outputs  |         4 |         4 |         0 |
```

```
-----------------------------+-----------+-----------+-----------+
                             |        38 |        38 |         0 |
```

The following is an example of a delay annotation report that includes a separate count for constant arcs.

```
pt_shell> report_annotated_delay -constant_arcs

****************************
report: annotated_delay
****************************

                             |           |           |    NOT    |
                             |   Total   | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
cell arcs                    |        19 |        17 |         0 |
cell arcs (unconnected)      |         1 |         1 |         0 |
constant arcs                |           |         2 |         0 |
internal net arcs            |         3 |         3 |         0 |
constant arcs                |           |         0 |         0 |
net arcs from primary inputs |        11 |        11 |         0 |
constant arcs                |           |         0 |         0 |
net arcs to primary outputs  |         4 |         4 |         0 |
constant arcs                |           |         0 |         0 |
-----------------------------+-----------+-----------+-----------+
                             |        38 |        38 |         0 |
```

See Also

- read_sdf
- remove_annotated_delay
- reset_design
- set_annotated_delay

---

report_annotated_parasitics

Reports net parasitics back-annotated on the current design.

Syntax

string *report_annotated_parasitics*

```
[-check]
[-no_check]
[-internal_nets]
[-boundary_nets]
[-driverless_nets]
[-loadless_nets]
[-pin_to_pin_nets]
```

Feedback

```
[-max_nets num]
[-list_annotated]
[-list_not_annotated]
[-list_lumped]
[-constant_arcs]
[-via]
[-via_verbose]
[net_list]
```

Data Types

```
net_list    list
num         integer
```

Arguments

`-check`

Checks the design to verify that all annotated RC networks are complete. This option causes the *report_annotated_parasitics* command to check that all fanouts of each net connect through the RC network to each driver of the net. An error message reports nets with incomplete RC networks. This option is the default for PrimeTime version C-2009.06 and later.

`-no_check`

Prevents checking of the design to verify that all annotated RC networks are complete.

`-internal_nets`

Reports only parasitics annotated on internal nets (nets connected only to cell pins).

`-boundary_nets`

only parasitics annotated on boundary (port) nets. A connection to any port qualifies a net as a boundary net.

`-driverless_nets`

Reports parasitics annotated on driverless nets (nets that do not have a global driver).

`-loadless_nets`

Reports parasitics annotated on loadless nets (nets that do not have a global load) are reported. Note that if there are nets that neither have a global driver nor a global load, those nets are counted as driverless nets and not as loadless nets for the report.

`-ignore_partially_annotated`

Ignores incomplete RC network annotations considering them as not annotated.

`-pin_to_pin_nets`

Reports parasitics annotated on nets that have at least one global driver and at least one global load pin.

`-max_nets` *num*

Limites reporting to the specified maximum number of nets for the *-list_annotated*, *-list_not_annotated* and *-list_lumped* options.

`-list_annotated`

Lists back-annotated nets. You cannot use this option together with the *-list_not_annotated* or *-list_lumped* options.

`-list_not_annotated`

Lists nets that are not back-annotated. You cannot use this option together with the *-list_annotated* or *-list_lumped* options.

`-list_lumped`

Lists lumped nets. You cannot use this option together with the *-list_annotated* or *-list_not_annotated* options.

`-constant_arcs`

Keeps a separate count for nets connected to pins with arcs that are disabled due to logic constants or case analysis. With this option, the *-list_annotated* or *-list_not_annotated* option does not list the constant nets. The total count at the end of the table includes the count of constant nets, and a count of constant nets suppressed (due to *-max_nets*) is shown separately from the count of ordinary nets suppressed.

`-via`

Report via annotation details.

`-via_verbose`

Report via annotation details in verbose.

net_list

Reports only the specified nets.

Description

This command reports nets annotated with parasitics in the current design. This command can also check the consistency of parasitics after reading ascii or binary parasitics. The report summarizes how many nets are annotated with reduced parasitics (pi models) or detailed parasitics (RC networks).

Feedback

Parasitics are associate with global nets. This report counts nets without considering hierarchical crossings - only one segment per global net is reported.

The report clearly shows how many nets are back-annotated with lumped, pi and detailed RC networks. The nets that are not annotated are shown in a separate column. In addition, nets that do not have a global driver and nets that do not have any global load pins are shown separately. If a net falls into both these categories, that is if a net does not have a driver and does not have a load, it is shown in driver-less nets row.

Details on which nets are annotated or not annotated can be displayed using either the *-list_annotated* or *-list_not_annotated* option. Listing annotated nets show a detailed description of RC networks. Listing unannotated nets show only a list of net names. By default, these reports only show 10 nets. To increase that limit, use the *-max_nets* option.

You can also report parasitics for a specific set of nets by using the *net_list* option. The *-list_annotated* and *-list_not_annotated* options restrict their scope to the nets in *net_list*.

When the *si_enable_analysis* variable is set, a column for coupled networks is added to the summary report that provides the number nets that have coupling capacitors.

If the *-check* option is used, and the command reports nets with incomplete parasitics, you can complete partially annotated nets using the *complete_net_parasitics* command.

Examples

The following is an example of an annotated parasitics report.

```
pt_shell> report_annotated_parasitics -check

*****************************
report: annotated_parasitics
        -check
        -internal_nets
        -boundary_nets
        -driverless_nets
        -loadless_nets
        -pin_to_pin_nets
*****************************

                    |         |         |         |    RC     |    Not    |
Net Type            |  Total  | Lumped  |  RC pi  | network |Annotated|
--------------------+---------+---------+---------+---------+---------+
Internal nets       |         |         |         |         |         |
  - Pin to pin nets |   23645 |       0 |       0 |   23644 |       1 |
  - Driverless nets |       3 |       0 |       0 |       0 |       3 |
  - Loadless nets   |       1 |       0 |       0 |       0 |       1 |
--------------------+---------+---------+---------+---------+---------+
Boundary/port nets  |         |         |         |         |         |
  - Pin to pin nets |      34 |       0 |       0 |      34 |       0 |
  - Driverless nets |       0 |       0 |       0 |       0 |       0 |
  - Loadless nets   |       0 |       0 |       0 |       0 |       0 |
```

```
--------------------+---------+---------+---------+---------+---------+
                    |   23683 |       0 |       0 |   23678 |       5 |
```

The following is an example of an annotated parasitics report for a coupled network.

```
pt_shell> report_annotated_parasitics -check

*****************************
report: annotated_parasitics
        -check
        -internal_nets
        -boundary_nets
        -driverless_nets
        -loadless_nets
        -pin_to_pin_nets
*****************************

                    |         |         |         |   RC    | Coupled |
 Not   |
Net Type            |  Total  | Lumped  |  RC pi  | network | network
 |Annotated|
--------------------+---------+---------+---------+---------+---------+--
-------+
Internal nets       |         |         |         |         |         |
       |
  - Pin to pin nets |   23645 |       0 |       0 |    5453 |   18191 |
    1 |
  - Driverless nets |       3 |       0 |       0 |       0 |       0 |
    3 |
  - Loadless nets   |       1 |       0 |       0 |       0 |       0 |
    1 |
--------------------+---------+---------+---------+---------+---------+--
-------+
Boundary/port nets  |         |         |         |         |         |
       |
  - Pin to pin nets |      34 |       0 |       0 |      12 |      22 |
    0 |
  - Driverless nets |       0 |       0 |       0 |       0 |       0 |
    0 |
  - Loadless nets   |       0 |       0 |       0 |       0 |       0 |
    0 |
--------------------+---------+---------+---------+---------+---------+--
-------+
                    |   23683 |       0 |       0 |    5465 |   18213 |
    5 |
```

The following is an example of an annotated parasitics report, with PrimeTime SI analysis off, showing constant nets.

```
pt_shell> report_annotated_parasitics -check -constant_arcs

*****************************
report: annotated_parasitics
```

```
        -check
        -internal_nets
        -boundary_nets
        -constant_arcs
        -driverless_nets
        -loadless_nets
        -pin_to_pin_nets
*****************************

                     |         |         |          |   RC     | Coupled |
 Not   |
Net Type             |  Total  | Lumped  |  RC pi   | network | network
 |Annotated|
---------------------+---------+---------+---------+---------+---------+--
-------+
Internal nets        |         |         |          |         |         |
       |
  - Pin to pin nets  |   23640 |       0 |       0 |    5453 |   18186 |
    1 |
  - Driverless nets  |       3 |       0 |       0 |       0 |       0 |
    3 |
  - Loadless nets    |       1 |       0 |       0 |       0 |       0 |
    1 |
  - Constant nets    |       5 |       0 |       0 |       0 |       5 |
    0 |
---------------------+---------+---------+---------+---------+---------+--
-------+
Boundary/port nets   |         |         |          |         |         |
       |
  - Pin to pin nets  |       6 |       0 |       0 |      12 |      22 |
    0 |
  - Driverless nets  |       0 |       0 |       0 |       0 |       0 |
    0 |
  - Loadless nets    |       0 |       0 |       0 |       0 |       0 |
    0 |
  - Constant nets    |       0 |       0 |       0 |       0 |       0 |
    0 |
---------------------+---------+---------+---------+---------+---------+--
-------+
                     |   23683 |       0 |       0 |    5465 |   18213 |
    5 |
```

See Also

- complete_net_parasitics
- read_parasitics
- remove_annotated_parasitics
- reset_design
- si_enable_analysis

Feedback

report_annotated_power

Reports annotated power.

Syntax

status *report_annotated_power*

```
[-list_annotated]
[-rails rail_list]
```

Data Types

rail_list	list

Arguments

`-list_annotated`

Reports a list of cells with power values annotated on them.

`-rails rail_list`

Reports the annotated power for the specified power supply nets (or rails in non-UPF mode). This feature is only valid if the *power_enable_multi_rail_analysis* variable is set to *true*.

Description

This command provides a summary report of annotated powers in the current design.

Examples

The following example shows how power is annotated, removed, and reported.

```
pt_shell> set_annotated_power -internal_power 1 -leakage_power 0.01 u0/*
1
pt_shell> report_annotated_power -list_annotated

****************************************
Report : annotated_power
        -list_annotated
****************************************

Annotated cell powers:
------------------------------
1.  u0/u0  (internal: 1  leakage: 0.01)
2.  u0/u1  (internal: 1  leakage: 0.01)

                              |           |           |    NOT    |
     Cell type                |    Total  | Annotated | Annotated |
------------------------------+-----------+-----------+-----------+
unresolved black-box cell     |        2  |        1  |        1  |
```

Feedback

```
leaf cell                     |         3  |         1  |         2  |
------------------------------+------------+------------+------------+
                              |         5  |         2  |         3  |

1
pt_shell> remove_annotated_power u0/u0
1
pt_shell> report_annotated_power

****************************************
Report : annotated_power
****************************************

                              |            |            |    NOT     |
     Cell type                |    Total   | Annotated  | Annotated  |
------------------------------+------------+------------+------------+
unresolved black-box cell     |         2  |         1  |         1  |
leaf cell                     |         3  |         0  |         3  |
------------------------------+------------+------------+------------+
                              |         5  |         1  |         4  |

1
```

See Also

- set_annotated_power
- remove_annotated_power

---

report_aocvm

Reports information about advanced on-chip variation (AOCV) derate tables and coefficients. Also displays details of path-based and graph-based AOCV calculation.

Syntax

```
status report_aocvm
   [-early]
   [-late]
   [-rise]
   [-fall]
   [-clock]
   [-data]
   [-voltage voltage_value]
   [-cell_delay]
   [-net_delay]
   [-list_annotated]
   [-list_not_annotated]
   [-coefficient]
   [-lib_cell]
```

```
   [-nosplit]
   [object_list]
```

Data Types

voltage_value	string
object_list	list

Arguments

`-early`

Displays only the early AOCV derate tables for the objects specified in the *object_list*. This argument can be used with the -list_not_annotated option.

`-late`

Displays only the late AOCV derate tables for the objects specified in the *object_list*. This argument can be used with the -list_not_annotated option.

`-rise`

Displays only the rise AOCV derate tables for the objects specified in the *object_list*. This argument can be used with the -list_not_annotated option.

`-fall`

Displays only the fall AOCV derate tables for the objects specified in the *object_list*. This argument can be used with the -list_not_annotated option.

`-clock`

Displays only the clock AOCV derate tables for the objects specified in the *object_list*. This argument can be used with the -list_not_annotated option.

`-data`

Displays only the data AOCV derate tables for the objects specified in the *object_list*. This argument can be used with the -list_not_annotated option.

`-voltage` *voltage_value*

Displays only the voltage value-specific AOCV derate tables for the objects specified in the *object_list*.

`-cell_delay`

Indicates that only cell delay AOCV derate tables are shown on the objects specified in the *object_list*.

`-net_delay`

Indicates that only net delay AOCV derate tables are shown on the objects specified in the *object_list*.

`-list_annotated`

Indicates that leaf cells and global nets that are annotated with AOCV derate tables are listed.

`-list_not_annotated`

Indicates that leaf cells and global nets that are not annotated with AOCV derate tables are listed.

`-coefficient`

Indicates that random AOCV coefficients are shown. You can specify a collection of lib_cell, lib_timing_arc, and cell objects in the *object_list* to display coefficients annotated only on those objects.

`-lib_cell`

This option can be used with either the -list_annotated or -list_not_annotated options to report the library cell name instead of the instance name.

`-nosplit`

Do not split lines when columns overflow.

`object_list`

Specifies either timing_path objects for which path-based AOCV metrics are to be reported; or timing arc objects for which graph-based AOCV metrics are to be reported; or design objects on which AOCV derate tables and advanced OCV derate coefficients have been annotated.

Description

This command displays the user-specified advanced on-chip variation (AOCV) information. The type of information displayed depends on the type of AOCV information annotated.

If the design has been annotated with AOCV derate tables using the *read_aocvm* command, the *report_aocvm* command outputs a summary showing the numbers of leaf cells and global nets annotated with AOCV derate tables. Lists of fully annotated, partially annotated, and not annotated leaf cells and global nets can be displayed using the *-list_annotated* and *-list_not_annotated* options. You can specify a collection of the design, library arcs, hierarchical cells, and nets in the *object_list* to display AOCV derate tables annotated only on those objects. The early, late, rise, fall, clock, and data flags can be used with the list_not_annotated option to filter the list of netlist objects reporting only those missing the specified derating types. If a netlist object is specified and the AOCV information has been read from a file, the full path to the file is listed.

If the design has been annotated with AOCV derate coefficients using the *set_aocvm_coefficient* command, the *report_aocvm -coefficient* command displays these

coefficients. You can specify a collection of library cells, library timing arcs, and cells in the *object_list* to display coefficients annotated only on those objects.

If a timing path is specified in the *object_list*, path-based metrics (distance, launch depth, and capture depth) for that path are displayed. Note that the depths shown in the report have been scaled by random AOCV coefficients, if they exist. Timing paths can be obtained using the *get_timing_paths* command. The path passed to *report_aocvm* has to be a path-based analysis path if a graph-based analysis path is specified an error is issued.

If a timing arc is specified in the *object_list*, graph-based metrics for that arc are displayed. Note that the depths shown in the report have been scaled by random AOCV coefficients, if they exist. Timing arcs can be obtained using the *get_timing_arcs* command. Graph-based timing arc metrics are only displayed when graph-based AOCV analysis has been enabled by setting the *timing_aocvm_enable_analysis* variable to *true*.

Examples

In the following example, the design has been annotated with AOCV derate factors using the *read_aocvm* command.

```
pt_shell> report_aocvm

****************************************
Report : aocvm
 ...
****************************************

             |           | Fully     | Partially | Not       |
             | Total     | annotated | annotated | annotated |
 ------------+-----------+-----------+-----------+-----------+
  Leaf cells |         6 |         0 |         4 |         2 |
  Nets       |         7 |         0 |         0 |         7 |
 ------------+-----------+-----------+-----------+-----------+
             |        13 |         0 |         4 |         9 |
```

The following example displays path-based metrics for the timing path specified in the *object_list*, for a design that has been annotated with AOCV deratings.

```
pt_shell> report_aocvm [get_timing_paths -pba_mode path \\
          -path_type full_clock_expanded]

****************************************
Report : aocvm
         object_list
 ...
****************************************

  Startpoint: ffL (rising edge-triggered flip-flop clocked by clk2)
  Endpoint: ffC (rising edge-triggered flip-flop clocked by clk2)
  Path Group: clk2
```

```
  Path metrics         Cell               Net
 ---------------------------------------------
  Distance           575.85            666.41
  Launch depth         2.69              3.07
  Capture depth        1.00              1.09
```

The following example displays graph-based metrics and AOCV derate factors for the timing arc specified in the *object_list*. This report is only displayed when graph-based AOCV is enabled. The arc depth and distance displayed is the most pessimistic depth and distance for all possible paths through this arc.

```
pt_shell> report_aocvm [get_timing_arcs -from u2/A -to u2/Z]

****************************************
Report : aocvm
         object_list
 ...
****************************************

  From pin: u2/A
  To pin:   u2/Z
  Arc type: cell (data network)

  AOCVM arc metrics      Launch
 ---------------------------------------
  Distance               702.14
  Depth                    3.00

  AOCVM arc derates      Launch
 ---------------------------------------
  Early rise             0.8705
  Early fall             0.8508
  Late rise              1.1357
  Late fall              1.1566
```

See Also

- get_timing_paths
- read_aocvm
- remove_aocvm
- report_timing
- set_aocvm_coefficient
- timing_aocvm_enable_analysis

Feedback

report_app_var

Shows the application variables.

Syntax

string *report_app_var*

```
[-verbose]
[-only_changed_vars]
[pattern]
```

Data Types

pattern	string

Arguments

`-verbose`

Shows detailed information.

`-only_changed_vars`

Reports only changed variables.

pattern

Reports on variables matching the pattern. The default is "*".

Description

The *report_app_var* command prints information about application variables matching the supplied pattern. By default, all descriptive information for the variable is printed, except for the help text.

If no variables match the pattern, an error is returned. Otherwise, this command returns the empty string.

If the *-verbose* option is used, then the command also prints the help text for the variable. This text is printed after the variable name and all lines of the help text are prefixed with "#".

The Constraints column can take the following forms:

{val1 ...}

The valid values must belong to the displayed list.

val \<= a

The value must be less than or equal to "a".

val \>= b

The value must be greater than or equal to "b".

b \<= val \<= a

The value must be greater than or equal to "b", and less than or equal to "a".

Examples

The following are examples of the *report_app_var* command:

```
prompt> report_app_var sh*
Variable                 Value     Type    Default   Constraints
------------------------ --------- ------- ---------
 ---------------------
sh_continue_on_error     false     bool    false
sh_script_stop_severity  none      string  none      {none W E}
 \.\.\.


prompt> report_app_var sh* -verbose
Variable                 Value     Type    Default    Constraints
------------------------ --------- ------- ---------
 ---------------------
sh_continue_on_error     false     bool    false
   # Allows source to continue after an error
sh_script_stop_severity  none      string  none      {none W E}
   # Indicates the error message severity level which would cause
   # a script to stop executing before it completes
 \.\.\.
```

See Also

- get_app_var
- set_app_var
- write_app_var

---

report_attribute

Reports the attributes of objects in the design.

Syntax

string *report_attribute*

```
[-class class_name]
[-nosplit]
[-application]
[-attributes attribute_list]
```

Feedback

```
[-summary]
[-format csv]
[-output file_name]
object_list
```

Data Types

```
class_name          string
attribute_list      list
style               string
file_name           string
object_list         list
```

Arguments

`-class class_name`

Specifies the type of object: *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, or *lib_pin*. Use this option only when the *object_list* is specified as a list of names, not as a collection.

`-nosplit`

Prevents the splitting of long lines in the report.

`-application`

Lists application attributes as well as user-defined attributes. Without this option, the command reports only user-defined attributes (those created by the *define_user_attribute* command).

`-attributes attribute_list`

Reports only the specified list of attributes for each object. Without this option, the command reports all user-defined attributes, plus all application attributes if the *-application* option is used.

`-summary`

Generates a summary report in column format showing multiple attributes on each line, one line per object. Without this option, the report uses multiple lines per object. You can use this option only with the *-attributes* option.

`-format csv`

Specifies the reporting format. The only allowed value is *csv*, which generates the report as a comma-separated value list. Without this option, the report is generated in table format. To write out a file that can be read into a spreadsheet, use the *-format csv* option with the *-output* option.

`-output file_name`

Writes out the report to a file instead of the screen.

object_list

Specifies the objects whose attributes are reported, either as a collection gathered by a get_* command (*get_cells*, *get_nets*, and so on) or a list of object names. If you specify a list of names, you must also use the *-class* option to specify the object type.

Description

The *report_attribute* command reports the attribute settings of specified objects in the design. To see a list of object attributes that you can query, use the *list_attributes* command. For descriptions of the attributes, see the man page for the object type, for example, *man cell_attributes* or *man net_attributes*.

By default, the command reports only user-defined attributes (those created by the *define_user_attribute* command). To also report application attributes (predefined by the PrimeTime application), use the *-application* option.

An attribute value can be an integer, floating-point number, Boolean value, text string, or collection. For a collection-type attribute, the *report_attribute* command reports only the name of the first object in the collection. To gather the attribute as a collection, use the *get_attribute* command.

Hierarchical attribute queries (attribute queries of the form: *attribute1.attribute2*) are supported in report_attribute and *-summary* except for collections containing lib (lib_pin, lib_cell etc) object types and also not in distributed hierarchical analysis.

To limit the report to a specific list of attributes, use the *-attributes* option.

Examples

The following example defines an integer attribute num_x for cells, sets the value on two cells, and reports the result.

```
pt_shell> define_user_attribute -type int -class cell num_x
pt_shell> set_user_attribute [get_cells {U137 U138}] num_x 30
Set attribute 'num_x' on 'U137'
Set attribute 'num_x' on 'U138'

pt_shell> report_attribute [get_cells {U137 U138}]
 ...
Design       Object        Type      Attribute Name      Value
-------------------------------------------------------------------
ORCA         U137          int       num_x               30

ORCA         U138          int       num_x               30
```

The following example reports application attributes as well as user-defined attributes.

```
pt_shell> report_attribute -application [get_cells U137]
 ...
```

```
Design       Object        Type      Attribute Name           Value
-------------------------------------------------------------------------
ORCA         U137          int       num_x                    30
ORCA         U137          float     area                     3.500000
ORCA         U137          string    base_name                U137
ORCA         U137          boolean   disable_timing           true
ORCA         U137          float     early_fall_cell_check_derate_factor
                                                              1.000000
ORCA         U137          float     early_fall_clk_cell_derate_factor
                                                              0.900000
 ...         ...           ...       ...                      ...
```

The following example reports details about the timing arcs in a cell.

```
pt_shell> report_attribute [get_timing_arcs -of_objects [get_cells U137]]
 \\
 -attributes {from_pin to_pin is_disabled delay_max_rise delay_min_rise}
 ...
Design      Object        Type      Attribute Name           Value
-------------------------------------------------------------------------
ORCA        arc           collection
                                    from_pin                 U137/A1
ORCA        arc           collection
                                    to_pin                   U137/Z
ORCA        arc           boolean   is_disabled              true
ORCA        arc           float     delay_max_rise           0.000000
ORCA        arc           float     delay_min_rise           0.000000
ORCA        arc           collection
                                    from_pin                 U137/A2
ORCA        arc           collection
                                    to_pin                   U137/Z
ORCA        arc           boolean   is_disabled              false
ORCA        arc           float     delay_max_rise           0.876045
ORCA        arc           float     delay_min_rise           0.876045
```

The following example is the same as the previous one, but in summary format.

```
pt_shell> report_attribute [get_timing_arcs -of_objects [get_cells U137]]
 \\
 -attributes {from_pin to_pin is_disabled delay_max_rise delay_min_rise}
 \\
 -summary
 ...
Object    from_pin    to_pin     is_disabled  delay_max_rise
 delay_min_rise
-------------------------------------------------------------------------
arc       U137/A1     U137/Z     true         0.000000       0.000000
arc       U137/A2     U137/Z     false        0.876045       0.876045
```

The following example is the same as the previous one, but in CSV format.

Feedback

```
pt_shell> report_attribute [get_timing_arcs -of_objects [get_cells U137]]
 \\
 -attributes {from_pin to_pin is_disabled delay_max_rise delay_min_rise}
 \\
 -format csv
 ...
Design,Object,Type,Attribute Name,Value
ORCA,arc,collection,from_pin,U137/A1
ORCA,arc,collection,to_pin,U137/Z
ORCA,arc,boolean,is_disabled,true
ORCA,arc,float,delay_max_rise,0.000000
ORCA,arc,float,delay_min_rise,0.000000
ORCA,arc,collection,from_pin,U137/A2
ORCA,arc,collection,to_pin,U137/Z
ORCA,arc,boolean,is_disabled,false
ORCA,arc,float,delay_max_rise,0.876045
ORCA,arc,float,delay_min_rise,0.876045
```

See Also

- collections
- define_user_attribute
- get_attribute
- list_attributes
- remove_user_attribute
- set_user_attribute
- timing_report_attribute_skip_table_header

---

report_bottleneck

Reports timing bottleneck information.

Syntax

string *report_bottleneck*

```
[-cost_type path_count | path_cost | fanout_endpoint_cost]
[-delay_type max | min]
[-slack_lesser_than slack_limit]
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-through through_list]
```

Feedback

```
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-max_cells cell_count]
[-max_paths path_count]
[-nworst_paths paths_per_endpoint]
[-group group_name]
[-significant_digits digits]
[-nosplit]
```

Data Types

slack_limit	float
from_list	list
rise_from_list	list
fall_from_list	list
to_list	list
rise_to_list	list
fall_to_list	list
through_list	list
rise_through_list	list
fall_through_list	list
cell_count	int
path_count	int
paths_per_endpoint	int
group_name	list
digits	int

Arguments

`-cost_type path_count | path_cost | fanout_endpoint_cost`

Specifies a cost type to use for computing the bottleneck cost:

- *path_count* (the default) - Uses the number of violating paths through the cell.
- *path_cost* - Uses the total slack of violating paths through the cell.
- *fanout_endpoint_cost* - Uses the total cost of violating endpoints in the fanout of the cell.

`-delay_type max | min`

Specifies whether to compute the bottleneck cost for

- *max* (the default) - Setup analysis.
- *min* - Hold analysis.

`-slack_lesser_than slack_limit`

Considers only those paths with a slack less than *slack_limit* to find bottlenecks.

`-from` *`from_list`*

Considers only paths from the specified pins, ports, nets, or startpoints clocked by named clocks to find bottlenecks.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_from` *`fall_from_list`*

Same as the *-from* option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-to` *`to_list`*

Considers only paths to the specified pins, ports, nets, or endpoints clocked by specified clocks.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

`-fall_to` *`fall_to_list`*

Same as the *-to* option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-through` *`through_list`*

Considers only paths through the specified pins, ports, nets or cells to find bottlenecks.

You can use multiple *-through*, *-rise_through*, and *-fall_through* options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one *-through*, *-rise_through*, or *-fall_through* option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

`-rise_through` *`rise_through_list`*

Same as the *-through* option, except that the path must rise through the specified objects.

`-fall_through` *`fall_through_list`*

Same as the *-through* option, except that the path must fall through the specified objects.

`-max_cells` *`cell_count`*

Specifies the number of cells to report. The default is 20.

`-max_paths` *`path_count`*

Specifies the number of paths to be considered per path group. Allowed values are 1 to 2000000; the default is to use the *paths_per_endpoint* setting.

`-nworst_paths` *`paths_per_endpoint`*

Specifies the number of paths to be considered per endpoint. Allowed values are 1 to 2000000; the default is 100.

`-group` *`group_name`*

Indicates that only paths in *group_name* are to be considered.

`-significant_digits` *`digits`*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default is 2. Use this option if you want to override the default.

`-nosplit`

Prevents line-splitting and facilitates writing software to extract information from the report output.

Description

This command reports information about timing bottlenecks in the current design. A bottleneck is a common point in the design that contributes to multiple violations.

To report bottlenecks throughout the current design, the arrival totals and slacks must be available on all pins, not just at endpoints. To achieve this, set the

timing_save_pin_arrival_and_slack variable to *true* before using this command. If the *timing_save_pin_arrival_and_slack* variable is set to *false*, this command automatically sets it to *true* and updates the design timing before the command executes.

If you intend to use this command, it is recommended that you set the *timing_save_pin_arrival_and_slack* variable to *true* before the first timing update, therefore preventing the cost of an additional timing update.

Examples

The following example reports the 20 worst bottleneck cells in the design based on the number of paths through the cell with slack less than 0.0.

```
pt_shell> report_bottleneck

****************************************
Report : bottleneck
         -max_cells 20
         -nworst_paths 100
Design : example
 ...
****************************************

Bottleneck Cost = Number of violating paths through cell

                                                  Bottleneck
Cell                                Reference     Cost
-------------------------------------------------------------------
ipxr/ir1/i2/i0/u3                   DFF1A         39656.00
ipxr/ir1/U14                        INV2          39654.00
ipxr/ir1/U16                        INV8          39654.00
ipxr/ir1/i2/i0/u4                   DFF1A         31806.00
ipxr/ir1/U15                        BUF8B         31804.00
ipxr/U1712                          MUX2I         23999.00
ipxr/U558                           INV4          23999.00
ipxr/U1370                          NAN4          22485.00
ipxr/x01                            INV2          22485.00
dmac/BufEn                          BUF3          22485.00
dmac/U574                           INV           22485.00
ipxr/U1335                          NAN6CH        21844.00
ipxr/U564                           MUX2I         20074.00
ipxr/U1694                          MUX2A         19936.00
ipxr/U1559                          BUF2C         14785.00
ipxr/U1484                          MUX2I         14252.00
ipxr/U1486                          MUX2I         12468.00
SccMOD/U929                         OR3           12135.00
ipxr/U1493                          MUX2I         11248.00
dmac/BiuSMOD/U879                   NAN2          10949.00
```

Feedback

See Also

- report_cell
- report_timing
- report_default_significant_digits
- timing_save_pin_arrival_and_slack

report_bounding_box

Reports the approximate bounding box of specified nets.

Syntax

status *report_bounding_box*

`[-of_objects nets]`

Data Types

nets	list

Arguments

`-of_objects nets`

A list of one or more nets for which to report bounding-box information. At least one net is required. Glob-style wildcards (*) are supported.

Description

This command reports the approximate bounding box of specified nets. The coordinates of ground capacitor nodes are used for computation.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a bounding box report.

```
pt_shell> report_bounding_box -of_objects {SUM0 B0}
========        ===========    =========    ===========    =========
Net Name        llx            lly          urx            ury
========        ===========    =========    ===========    =========
SUM0            -467.000000    11.000000    -458.000000    82.000000
B0              -497.000000    2.500000     -272.000000    82.000000
```

report_budget

Reports the budget allocation calculations for a selected timing path.

Feedback

Syntax

string *report_budget*

```
-through      pin_name
[-delay_type  delay_type]
[-show_slack]
[-nosplit]
```

Data Types

delay_type	string
pin_name	string

Arguments

`-through` *pin*

Reports the budget allocation of worst timing path through the given pin.

The pin must be a hierarchical pin. The *update_budget* command must be run prior to using this command.

`-delay_type` *delay_type*

Specifies the type of path delay constraint to consider when reporting worst-slack budgeted paths.

Valid values are: *min*, *max*, *min_rise*, *min_fall*, *max_rise*, *max_fall*.

`-show_slack`

Shows slack after budgeting. This option prints the following rows below the budget report:

- "Block slack at pin" - shows the budget-estimated slack for the block-level analysis at the specified *-through* pin, assuming that the budgeted constraints are used.
- "Top segment slack" - shows the budget-estimated slack for the top-level analysis at the specified *-through* pin.

`-nosplit`

Prevents line splitting. This can useful for scripts that extract information from the report. By default, the report generates a new line when the text cannot fit in the alloted space in a column.

Feedback

Examples

The following command reports the budget allocation of the worst path that goes through hierarchical pin mid/i1:

```
pt_shell> report_budget -through mid/i1
 ...

Slack:  -5.23
Required time:   5.00
From clock: clk_v2
To clock: clk_r
Path Type: max fall

Point                          Mode                      Delay     Weight
 Budget
-------------------------------------------------------------------------
-------
i1
mid/i1                         pin_slack                   0.27     0.38
 -1.72
mid/ff1/D                      pin_slack                   0.44     0.62
 -2.79

mid/i1 -> mid/ff1/D
  Budget = delay + slack * weight =   0.44 +  -5.23 *   0.62 =  -2.79
  Input delay margin = slack + delay - budget =  -5.23 +   0.44 -  -2.79
 =  -1.99
  Input budget = context + Input delay margin =   8.28
```

See Also

- set_timing_budget
- update_budget

---

report_bus

Reports the bused ports or nets in the current instance or current design.

Syntax

string *report_bus*

```
[-nosplit]
```

Arguments

`-nosplit`

Does not split lines if column overflows.

Feedback

Description

Displays information about buses (pins or nets) in the current instance or current design. If the current instance is set, the report is generated for the design of that instance; otherwise, the report is generated for the current design.

Examples

The following command reports the buses in the design.

```
pt_shell> report_bus

****************************************
Report : bus
Design : new_design
****************************************

Bussed Port      Dir      From     To     Width
------------------------------------------------------------
acnt             out      15       0      16
baddreg          out      15       0      16
bwordreg         out      15       0      16
caddreg          out      15       0      16
```

See Also

- report_port

---

report_case_analysis

Reports case analysis entries on ports and pins, and the propagation of user-specified case values and logic constant values throughout the netlist.

Syntax

```
status report_case_analysis
  [-all]
  [-nosplit]
  [-to]
  [-from]
  [-sequential_propagation]
  [pin_or_port_list]
```

Data Types

```
pin_or_port_list      list
```

Feedback

Arguments

`-all`

Reports all pins that have set case analysis values and reports the built-in constant pins of the design that are considered for startpoints of logic constant propagation. Logic constant propagation is performed by default from the constant pins of the design, unless the *disable_case_analysis* variable is set to *true*.

`-nosplit`

Prevents line splitting to facilitate parsing of information from the report output.

`-to`

Shows the backward trace of case propagation to each pin or port specified by *pin_or_port_list*. The backward trace expands only to pins or ports with case values that contribute to the case value on the destination pin or port. If the number of pins or ports involved in backward trace exceeds the number specified by *-max_objects*, PrimeTime truncates the trace and prints a message at the end of the trace to indicate an incomplete trace.

`-from`

Shows the forward trace of case propagation from each pin or port specified by *pin_or_port_list*. The forward trace expands only to pins or ports where the case value from the source contributes to the case value on that pin or port. If the number of pins or ports involved in the forward trace exceeds the number specified by *-max_objects*, PrimeTime truncates the trace and prints a message at the end of the trace to indicate an incomplete trace.

`-sequential_propagation`

Reports sequential cell instances and library cells that are enabled for constant propagation. To enable constant propagation on specific sequential cells, set the *case_analysis_sequential_propagation* variable to *never*, and run the *set_case_sequential_propagation* command.

`pin_or_port_list`

Reports the propagated case values on the specified pins or ports.

Description

This command reports case analysis settings on ports and pins that are specified by the *set_case_analysis* command.

If you do not specify *pin_or_port_list*, the report lists all pins and ports on which case analysis is set; this report does not show where the logic constant values are propagated.

Feedback

Examples

The following example specifies that pins U1/U2/A and U1/U3/CI are set to a constant logic I1:

```
pt_shell> set_case_analysis 1 {U1/U2/A U1/U3/CI}
pt_shell> report_case_analysis

****************************************
Report : case_analysis
 ...
****************************************

Pin name                     User case analysis value
-----------------------------------------------------
U1/U2/A                       1
U1/U3/CI                      1
```

The following example displays the fanout trace of the case value on port CL:

```
pt_shell> report_case_analysis -from CL

****************************************
Report : case_analysis
Design : counter
 ...
****************************************
Properties         Value   Pin/Port
----------------------------------------
user case          1       CL


Case fanout report:
Verbose Forward Trace for pin/port CL:

Pin/Port ID  Value  Fanout Pin/Port IDs  Pin/Port
-------------------------------------------------
0            1      1 8                 CL
1            1      3                   p/B
2            1      3                   p/A
3            1      4 5 6 7             p/Z
4            1                          a/D
5            1                          j/D
6            1                          g/D
7            1                          d/D
8            1                          o/A
9            0                          o/B
```

The following example enables sequential propagation on cell instances (FF1, FF2, and FF3), library cells (D_FF, D_FF_R, and D_FF_S) and then prints them using the '-sequential_propagation' option in the report_case_analysis command.

Feedback

```
pt_shell>set_case_sequential_propagation { FF1 FF2 FF3 }
pt_shell>report_case_analysis -sequential_propagation
****************************************
Report : case_analysis
         -sequential_propagation
 ...
****************************************

--------------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 3
--------------------------------------------------------------------
Cell Instance : 'FF1'
Cell Instance : 'FF2'
Cell Instance : 'FF3'
--------------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 3
--------------------------------------------------------------------
Lib Cell : 'snps_test/D_FF_R'
Lib Cell : 'snps_test/D_FF'
Lib Cell : 'snps_test/D_FF_S'
```

See Also

- remove_case_analysis
- remove_case_sequential_propagation
- set_case_analysis
- set_case_sequential_propagation
- disable_case_analysis

---

report_cell

Reports cell information.

Syntax

status *report_cell*

```
[-connections]
[-verbose]
[-nosplit]
[-significant_digits digits]
[cell_list]
```

Data Types

```
digits        integer
cell_list     list
```

Feedback

Arguments

`-connections`

Displays the pins and the nets to which they connect.

`-verbose`

Displays detailed connection information. For each input pin, displays the net and the driver pins on that net. For each output pin, displays the net and the load pins on that net. Use this option together with the *-connections* option.

`-nosplit`

Prevents line splitting in the report when a field exceeds the column width.

`-significant_digits` *`digits`*

Specifies the number of digits to the right of the decimal point shown in the report. Allowed values are 0-13. The default is determined by the *report_default_significant_digits* variable, which has a default of 2. Use this option if you want to override the default. Because the tool uses fixed-precision floating-point arithmetic for delay calculation, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h. The upper bound on a meaningful value of the argument to *-significant_digits* is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

`cell_list`

Specifies the cells to report. If you do not specify this option, all cells in the current instance are listed.

Description

This command displays information and statistics about cells in the current instance or current design. If you set for the current instance, the report is generated for the design of that instance. Otherwise, the report is generated for the current design.

If any cell uses nondefault operating conditions, the *report_cell* command shows information about the operating conditions for all cells.

Some information, such as whether the cell is in an ideal network or not, is displayed after a timing update (as a result of manually executing the *update_timing* command function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about cells without incurring the cost of a complete timing update.

Examples

The following example displays a report of the cells in the current design.

```
pt_shell> report_cell

****************************************
Report : cell
 ...
****************************************

Attributes:
    b - black-box (unknown)
    h - hierarchical
    n - noncombinational
    u - contains unmapped logic
    A - abstracted timing model
    E - extracted timing model
    S - Stamp timing model
    Q - Quick timing model (QTM)
    I - ideal network

Cell                      Reference       Library             Area
 Attributes
--------------------------------------------------------------------------
-------
i1                        inter                                6.00  h
low_i                     low                                  2.00  h
n0_i                      ND2             my_lib               1.00
o_reg2                    FD2             my_lib               9.00  n
--------------------------------------------------------------------------
-------
Total 4 cells                                                 18.00
```

The following example gives a verbose report of the cell connections.

```
pt_shell> report_cell -verbose -connections

****************************************
Report : cell
        -connections
        -verbose
 ...
****************************************

Connections for cell 'i1':
    Reference:          inter
    Hierarchical:       TRUE
    Area:               6

    Input Pins          Net            Net Driver Pins   Driver Pin Type
    ----------------    ------------   ----------------  ----------------
    a                   out_1          o_reg1/Q          Output Pin (FD2)
    b                   out_2          o_reg2/Q          Output Pin (FD2)
    c                   out_3          o_reg3/Q          Output Pin (FD2)
    d                   out_4          o_reg4/Q          Output Pin (FD2)
```

```
    Output Pins          Net             Net Load Pins      Load Pin Type
    ----------------     ------------    ----------------   ----------------
    z1                   i1t1            low_i/n1/A         Input Pin (NR4)
    z2                   i1t2            low_i/n1/B         Input Pin (NR4)
    z3                   i1t3            low_i/n1/C         Input Pin (NR4)

Connections for cell 'low_i':
    Reference:           low
    Hierarchical:        TRUE
    Area:                2

    Input Pins           Net             Net Driver Pins    Driver Pin Type
    ----------------     ------------    ----------------   ----------------
    a                    i1t1            i1/low/n1/Z        Output Pin (NR4)
    b                    i1t2            i1/low2/n1/Z       Output Pin (NR4)
    c                    i1t3            i1/low3/n1/Z       Output Pin (NR4)
    d                    in2             in2                Input Port

    Output Pins          Net             Net Load Pins      Load Pin Type
    ----------------     ------------    ----------------   ----------------
    z                    low             o_reg2/D           Input Pin (FD2)

Connections for cell 'n0_i':
    Reference:           ND2
    Library:             my_lib
    Area:                1

    Input Pins           Net             Net Driver Pins    Driver Pin Type
    ----------------     ------------    ----------------   ----------------
    A                    p_in            p_in               Input Port
    B                    p_in            p_in               Input Port

    Output Pins          Net             Net Load Pins      Load Pin Type
    ----------------     ------------    ----------------   ----------------
    Z                    n0              n1_i/B             Input Pin (ND2)

Connections for cell 'o_reg2':
    Reference:           FD2
    Library:             my_lib
    Area:                9

    Input Pins           Net             Net Driver Pins    Driver Pin Type
    ----------------     ------------    ----------------   ----------------
    D                    low             low_i/n1/Z         Output Pin (NR4)
    CP                   CLOCK           CLOCK              Input Port
    CD                   OPER            OPER               Input Port

    Output Pins          Net             Net Load Pins      Load Pin Type
    ----------------     ------------    ----------------   ----------------
    Q                    out_2           i1/low3/n1/B       Input Pin (NR4)
                                         out_2              Output Port
                                         i1/low/n1/B        Input Pin (NR4)
                                         i1/low2/n1/B       Input Pin (NR4)
```

```
    QN                  NET2QNX        i2/low3/n1/B      Input Pin (NR4)
                                       i2/low/n1/B       Input Pin (NR4)
                                       i2/low2/n1/B      Input Pin (NR4)
```

In the following example, cell U2 uses nondefault operating conditions. Therefore, the *report_cell* command shows information about the operating conditions for all cells.

```
pt_shell> report_cell "B/C/u1  B/C/D/E/u2 B/C/D/E/u3 "

****************************************
Report : cell
 ...
****************************************

Attributes:
    b - black-box (unknown)
    h - hierarchical
    n - noncombinational
    u - contains unmapped logic
    A - abstracted timing model
    E - extracted timing model
    S - Stamp timing model
    Q - Quick timing model (QTM)
    I - ideal network

Cell          Reference       Library          Area  Attributes
------------------------------------------------------------------------
u1            BUFFD1          tech90           3.23
u2            BUFFD1          tech90           3.23
u3            BUFFD1          tech90           3.23
------------------------------------------------------------------------
Total 3 cells                                  9.68

Cell specific operating conditions and rail voltages:

         Min             Max             Min      Max
Cell     OC              OC              Voltage  Voltage
------------------------------------------------------------------
u1       tech90/WCCOM    tech90/WCCOM
u2       tech90/new_oc   tech90/new_oc
u3       tech90/WCCOM    tech90/WCCOM
```

See Also

- current_design
- current_instance
- report_design
- report_hierarchy

Feedback

- report_net
- report_port
- report_reference

report_cell_em_violation

Reports the toggle rate violations for specific cells or pins.

Syntax

status *report_cell_em_violation*

```
[-cell cell_list]
[-pin pin_list]
[-exclude_arc]
[-nworst number]
[-current_type type]
[-verbose]
[-all]
```

Data Types

cell_list	list
pin_list	list
number	integer
type	string

Arguments

`-cell cell_list`

Reports toggle rate violations for the pins of the cells specified in the *cell_list*. The *-cell* and *-pin* options are mutually exclusive.

`-pin pin_list`

Reports toggle rate violations for the pins specified in the *pin_list*. The *-cell* and *-pin* options are mutually exclusive.

`-exclude_arc`

Prevents the use of arc-based data for calculations. If you do not use this option, the command uses all table data (arc-based and non-arc-based) for toggle rate violation calculations.

`-nworst number`

Reports the specified number of cells or pins with the worst toggle rate violations.

Feedback

`-current_type` *type*

Specifies the table to use for toggle rate violation calculations. If the switch is not specified, the command checks all three current types and report the worst one .

`-verbose`

Shows the slew rate and cap values used for toggle rate violation calculations. Also reports errors for missing electromigration data for pins.

`-all`

Shows all pins. If you do not use this option, by default, the command reports only the pins with toggle rate violations.

Description

This command reports the toggle rate violations for the cells or pins in the design.

Examples

The following command reports violations for all pins in the design with toggle rate violations.

```
pt_shell> report_cell_em_violation
```

The following command reports all pins with and without toggle rate violations. Pins with toggle rate violations will have violation metrics.

```
pt_shell> report_cell_em_violation -pin [get_pins] -all
```

The following command reports the violations for all pins with toggle rate violations in the design. Arc-based tables are ignored for the calculations.

```
pt_shell> report_cell_em_violation -pin [get_pins] -exclude_arc
```

See Also

- get_em_max_toggle_rate

---

report_cell_mode

Reports the modes for specified cells.

Syntax

```
status report_cell_mode
   [-nosplit]
   [instance_list]
```

Feedback

Data Types

instance_list list

Arguments

-nosplit

Prevents line splitting when report columns overflow.

instance_list

Specifies a list of instances whose modes are to be reported. By default, the report includes all cells that have modes. This option is only valid for the *-type cell* option.

Description

This command reports cell modes. The report specifies whether the cell mode is enabled or disabled, and one of the following reasons for enabling or disabling the mode:

- cell - Indicates that the cell mode has been set directly using the *set_cell_mode* command.
- design - Indicates that the cell mode has been set due to the activation of a design mode using the *set_mode -type design* command. The design mode must have previously been mapped to the cell mode using the *map_design_mode* command. The report also specifies the name of the activated design mode. To view more details about the design mode, use the *report_mode -type design* command.
- cond - Indicates that the cell mode has been set due to evaluation of the mode condition during constant propagation.
- default - Indicates that the cell mode has not been set by any of the above reasons.

Examples

The following example reports cell mode information for the design top_design. Two cells have modes, Uram1 and Uram2 are instances of the library cell RAM2_core. rw is a cell mode group defined on the library cell RAM2_core. This cell mode group has two cell modes read and write. No modes are currently active so all modes are enabled and the reason is given as default.

```
pt_shell> report_cell_mode

****************************************
Report : mode
Design : top_design
****************************************

Cell                              Mode(Group)       Status      Reason
```

```
------------------------------------------------------------------------
---
Uram1/core(RAM2_core)              read(rw)          ENABLED     default
                                   write(rw)         ENABLED     default
------------------------------------------------------------------------
---
Uram2/core(RAM2_core)              read(rw)          ENABLED     default
                                   write(rw)         ENABLED     default
------------------------------------------------------------------------
---
```

The following example shows a report of modes specified for the two RAMs that have modes in the design. Ram Uram1 is set in mode read, and Ram Uram2 is set in mode write. Thus, all timing arcs of RAM Uram1 associated with mode read are enabled , and all timing arcs associated with mode write are disabled.

```
pt_shell> set_cell_mode read Uram1/core
pt_shell> set_cell_mode write Uram2/core
pt_shell> report_cell_mode

****************************************
Report : mode
Design : top_design
****************************************

Cell                               Mode(Group)     Status          Reason
------------------------------------------------------------------------
---
Uram1/core(RAM2_core)              read(rw)        ENABLED         cell
                                   write(rw)       disabled        cell
------------------------------------------------------------------------
---
Uram2/core(RAM2_core)              read(rw)        disabled        cell
                                   write(rw)       ENABLED         cell
------------------------------------------------------------------------
---
```

See Also

- map_design_mode
- define_design_mode_group
- remove_design_mode
- reset_cell_mode
- reset_mode
- set_cell_mode
- set_mode

Feedback

report_cell_robustness

Runs unified cell sensitivity analysis at the design level and reports a list of critical cells sensitive to specific parameters.

Syntax

integer *report_cell_robustness*

```
[-pba]
[-dvd]
[-verbose]
[-reg_to_reg]
[-max_violators num_cells]
[-nosplit]
[-significant_digits digits]
[-sort_by slack | delay]
[-type variation | voltage | transition | load_cap | wire_cap |
 {analysis_type_list}]
[-voltage_shift value]
[-voltage_shift_ratio value]
[-tran_shift_ratio value]
[-loadcap_shift_ratio value]
[-wirecap_shift_ratio value]
[-slack_lesser_than maximum_robustness_slack]
[-slack_margin margin_value]
[-mim_filter none | logical | timing]
[-cells cell_list]
[-net]
[-delay_type delay_type]
[-plot]
[-prefix string]
[-compare {file1 file2 ...}]
[-title title_string]
```

Data Types

num_cells	integer
digits	integer
analysis_type_list	list
value	float
maximum_robustness_slack	float
margin_value	float
cell_list	list
delay_type	string
file	string
title_string	string

Feedback

Arguments

`-pba`

Performs cell robustness using path-based timing analysis, where on-demand exhaustive path-based analysis (PBA) pin slacks are used to avoid pessimism. Without this option, cell robustness analysis uses graph-based analysis (GBA) and GBA pin slacks are used to evaluate the cell high-sigma failure rate.

`-dvd`

Performs voltage robustness analysis using the dynamic voltage drop (DVD) information read-in previously using the *read_dvd* command. The variable *timing_enable_dvd_analysis* must be set to *true* for the DVD voltage robustness analysis to be performed. An additional column “Delta_V” is reported in the voltage robustness report to show drop in voltage for each cell based on the DvD file. Please note that "-dvd" cannot be used with "-voltage_shift" or "-voltage_shift_ratio" options.

`-verbose`

Shows detailed information on robustness violators in the report.

`-max_violators` *`num_cells`*

Specifies the maximum number of cells to be reported. The default is 10000.

`-nosplit`

Prevents line-splitting to facilitate parsing of information from the report output. With this option, the design information is listed in fixed-width columns. Without this option, if the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`-significant_digits` *`digits`*

Specifies the number of digits after the decimal point displayed for the values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which is 2 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

`-reg_to_reg`

Restricts analysis to register-to-register paths. This option is supported for both GBA and PBA analyses.

`-sort_by slack | delay`

Specifies how the reported cells are sorted. The default is *slack*, which sorts by robustness slack. A value of *delay* sorts by delay impact instead. This option does not apply to variation robustness.

`-type variation | voltage | transition | load_cap | wire_cap | {`*`analysis_type_list`*`}`

Specifies the types of robustness analysis to be performed. One or more types from the following can be specified as a list. The default is *variation* (for backward compatibility with previous release). The supported types of analyses are:

- *variation* (the default) -- identify cells with larger POCV variation.
- *voltage* -- identify cells sensitive to voltage rail (IR) drop.

  This option makes *report_cell_robustness -type voltage* to be identical to *report_voltage_robustness* command.
- *transition* -- identify cells sensitive to input transition change.
- *load_cap* -- identify cells sensitive to output pin load capacitance (wire plus pin capacitance) changes.
- *wire_cap* -- identify cells sensitive to net wire capacitance changes (i.e. changes due to net parasitics).

The *analysis_type_list* must include types only from these supported types. When multiple types are specified, the analyses are performed concurrently. Separate reports will be generated for each analysis performed concurrently.

`-voltage_shift` *`value`*

Specifies the voltage drop used for the sensitivity calculation, as an absolute value in volts. The default is a 5% relative drop from each cell's nominal VDD value. This option applies only to voltage robustness.

`-voltage_shift_ratio` *`value`*

Specifies the voltage drop used for the sensitivity calculation, as a relative ratio (such as 0.95) of each cell's nominal VDD value. The default is a 5% relative drop. This option applies only to voltage robustness.

`-tran_shift_ratio` *`value`*

Specifies the transition change used for the sensitivity calculation, as a relative ratio (such as 0.10) of each cell's computed GBA transition time value. The default is a 10% relative increase. This option applies only to transition robustness.

`-loadcap_shift_ratio` *`value`*

Specifies the load cap change used for the sensitivity calculation, as a relative ratio (such as 0.10) of the cap of the net driven by the cell. The default is a 10% relative increase. This option applies only to load_cap robustness.

`-wirecap_shift_ratio` *value*

Specifies the wire cap change used for the sensitivity calculation, as a relative ratio (such as 0.10) of the cap of the net driven by the cell. The default is a 10% relative increase. This option applies only to wire_cap robustness.

`-slack_lesser_than` *maximum_robustness_slack*

Reports only cells with robustness slack less than the specified *maximum_robustness_slack* value; these cells have a negative robustness slack worse than the specified *maximum_robustness_slack* value (or a positive slack that is closer to causing a violation).

The default *maximum_robustness_slack* value for this option is 0.0.

`-slack_margin` *margin_value*

This option can be used to filter cells from the robustness report. The cells whose worst timing slack (GBA or PBA), amongst paths passing through the cell pins, is above the *margin_value* are filtered out from the report.

The default *margin_value* value for this option is infinity (that is, no filtering margin is applied).

`-mim_filter none | logical | timing`

Specifies how the reported cells are filtered out based on multiply intantiated modules (MIMs) grouping. This option can be used to increase the coverage in the report, given a *-max_violators* count. The default value is *none*, that is, no MIM grouping is considered. When *logical* is specified, only the instance with worst robustness slack, amongst all logically shared MIM instances, is included in the report. With *timing*, the worst, amongst all MIM instances that share delays, will be reported.

`-cells` *cell_list*

Specifies a list of one or more cells in the current design to be analyzed. The analysis is performed only on the cells in the list.

`-net`

Shows the net delay impact for wire cap and load cap robustness analyses. This option must be used with the *-verbose* option.

`-delay_type` *delay_type*

Specifies the type of timing and delay constraint to consider for performing the cell robustness analysis:

- *max* (default) -- max delay (setup constraint)
- *min* -- min delay (hold constraint)

`-plot`

Generates Gnuplot data and script file for the visualization of the robustness metrics.

`-prefix` *`string`*

Use with the *-plot* option. If a prefix string is provided using this option, this string is preprended to the generated Gnuplot file names.

`-compare {`*`file1 file2 ...`*`}`

Use only for the comparison of the robustness analysis result files. This option enables the plotting of data from multiple previous runs in a single graph. *file1*, *file2*, …. are robustness data files, that is, the .dat files generated using the *-plot* option, from previous analysis runs.

`-title` *`title_string`*

Use with the *-compare* option. If a title string is provided using this option, this string is used as the plot title for the comparison plot generated by the *-compare* option.

Description

The *report_cell_robustness* command can be used to report a list of cells sensitive to specific parameters. Use the command to run unified robustness analysis. Currently, the supported parameters to evaluate cell sensitivity are variation robustness, voltage robustness, transition robustness, load cap robustness and wire cap robustness. The default is variation analysis, which reports a list of cells with a high-sigma failure rate.

The variation robustness analysis of the *report_cell_robustness* command reports a list of cells with a high-sigma failure rate. The high-sigma failure rate is evaluated when a cell is at high-sigma, while other cells have normal POCV variation. The command supports robustness analysis using either GBA or PBA.

By default, GBA-based robustness analysis is performed and GBA pin slacks are used to evaluate the cell high-sigma failure rate. Using the *-pba* option invokes PBA-based robustness analysis to reduce pessimism, where exhaustive PBA pin slacks are computed on-the-fly and used for cell high-sigma failure rate evaluation.

The variation robustness analysis of the *report_cell_robustness* command is similar to the *report_variation_bottleneck* command, which also reports critical cells with high-sigma failure rate.

The voltage robustness analysis of the *report_cell_robustness* command is identical to the *report_voltage_bottleneck* command. Refer to man page of the *report_voltage_bottleneck* command for details.

Feedback

Transition robustness analysis identifies the cells in the design that are sensitive to transition changes. The *report_cell_robustness -type transition* command performs this transition robustness analysis.

You can specify the transition shift as a relative factor from the computed GBA pin transition. The default is a 10% relative increase from the GBA pin transition time.

To identify cells that are sensitive to capacitance changes, wire cap or load cap analyses can be performed with *report_cell_robustness -type wire_cap* or *report_cell_robustness -type load_cap* options respectively.

- The *report_variation_bottleneck* command requires a path collection and performs variation bottleneck analysis only for cells in the given paths.
- The *report_cell_robustness* command does not need a path collection. It calculates exhaustive PBA slack values on-demand and uses them to calculate the high-sigma failure rate of all critical cells in the design. Only setup slack analysis is supported in this command.

To perform cell robustness analysis, the *ps_enable_analysis* variable must be set to *true*. The *ps_enable_save_timing_slack_data* variable must also be set to *true* before the first timing update.

Examples

The following example shows a cell robustness report for variation analysis.

```
prompt> report_cell_robustness -significant_digits 6
 ...
   Cell      Lib_cell      Variation    Sigma_ratio    HSFR
                                                    (x 1e-3)
   ------------------------------------------------------------
   nd184     ND2D0XPD      1.107004     0.834039     0.031647
   an463     AN2D1XPD      2.319284     0.661067     0.031636
   nd205     ND2D0XPD      2.280516     0.665559     0.031624
   ff197     DFQD0APD      1.785426     0.604753     0.012661
   ff244     DFQD0BPD      1.638114     0.559612     0.008816
   ------------------------------------------------------------
   Total:                                           0.116385
```

The cell robustness report has the following columns:

- *Cell*: The cell instance with a critical high-sigma failure rate.
- *Lib_cell*: The library cell name of the instance.
- *Variation*: The cell delay variation, which is the cell arc POCV corner delay minus the nominal delay.

- *Sigma_ratio*: The ratio of cell delay sigma to pin slack sigma.
- *HSFR*: The high-sigma failure rate.

The end of the report shows the total high-sigma failure rate, which is the sum of the high-sigma failure rates for all cells.

Often a small number of cells contribute to a large part of the total high-sigma failure rate. Therefore, performing timing ECOs on a few cells at the top of the list can improve the overall joint success rate significantly.

The following example shows how to generate a voltage robustness report:

```
## Must save timing and arrival data
pt_shell> set ps_enable_analysis true
pt_shell> set ps_enable_save_timing_slack_data true

pt_shell> define_scaling_lib_group { ... }
pt_shell> set_voltage ...

pt_shell> update_timing -full

pt_shell> report_cell_robustness -type voltage -max_violators 1 \\
           -voltage_shift 0.05 -nosplit
```

The following output is a voltage robustness violation report for the previous example.

```
voltage_robustness
                                Timing    Delay     Robustness
   Cell              Lib_cell   slack     impact    slack
   ---------------------------------------------------------
   uA/uB/U38           NOR2    -3.5523    0.0321    -0.0321
```

The robustness report shows the delay impact and robustness slack for each violator. The report is sorted based on the robustness slack by default.

The "Delay impact" column shows the impact on cell delay when the voltage is shifted by the value specified.

```
Timing slack = slack at nominal VDD
Delay impact = (cell delay at shifted VDD) - (cell delay at nominal VDD)
```

The "Robustness slack" column shows the new slack with the adjusted cell delay.

```
Robustness slack = (slack at nominal VDD) - (Delay impact)
```

If the slack at nominal VDD is negative, then even cells with a small delay impact can be reported with negative robustness slack. This can mask voltage-drop bottleneck cells with high delay impact on marginally passing paths.

To avoid missing any true voltage-drop bottleneck cells in these cases, the tool uses *0* as the slack for negative-slack paths at nominal VDD, for the robustness slack calculation.

The following is an example of a transition robustness analysis report. It is similar to the voltage robustness report, except the parameter varied is the pin transition time.

```
pt_shell> report_cell_robustness -type transition -max_violators 2 \\
          -tran_shift 0.05 -nosplit

transition_robustness
                                 Timing     Delay      Robustness
   Cell              Lib_cell    slack      impact     slack
   ------------------------------------------------------------
   nd2               ND2D3LVT   -0.65076    0.00096    -0.00096
   or1               OR2D1LVT   -1.24974    0.00084    -0.00084
```

The robustness report shows the delay impact and robustness slack for each violator. The report is sorted based on the robustness slack by default.

The "Delay impact" column shows the impact on cell delay when transition is shifted by the value specified. The robustness slack calculation is identical to voltage robustness analysis.

Multiple robustness analyses can be performed concurrently. The report for each analysis is produced separately as if the analyses were run using separate commands. Concurrent analysis is most helpful when used with *-pba* option as the PBA pin slack calculation is performed only once (thereby speeding up the overall analysis as compared to when each analysis type is run separately). For example, the following command runs variaton and transition robustness analyses concurrently:

```
pt_shell> report_cell_robustness -type { variation transition } -pba
 -max_violators 2 \\
          -tran_shift 0.05 -nosplit
```

See Also

- ps_enable_analysis
- get_timing_paths
- get_design_variation
- report_design_variation
- report_variation_bottleneck
- report_voltage_robustness
- ps_enable_save_timing_slack_data
- timing_save_pin_arrival_and_required
- timing_save_pin_arrival_and_slack

Feedback

- read_dvd
- remove_dvd

report_cell_usage

Reports the usage of cells in a design.

Syntax

string *report_cell_usage*

```
[-pattern_priority pattern_list]
[-attribute attribute_name]
[-alternative_lib_cells]
[-show_others]
[-power_attribute power_attribute_name]
```

Data Types

attribute_name	string
pattern_list	list
power_attribute_name	string

Arguments

`-pattern_priority` *pattern_list*

Specifies a list of library cell patterns, from highest to lowest priority.

`-attribute` *attribute_name*

Specifies an attribute name that can be used either to match patterns specified by the *-pattern_priority* option or to specify power attribute of each library cell. When you specify this option together with the *-pattern_priority* option, PrimeTime uses the value of the attribute instead of library cell names for pattern matching. When you specify this option without the *-pattern_priority* option, PrimeTime uses the value of the attribute as power attribute of each library cell.

`-alternative_lib_cells`

Shows alternative library cells. This option will be obsolete in a future release; use the *report_eco_library_cells* command instead.

`-show_others`

Shows library cells that do not match the specified pattern. This option will be obsolete in a future release; use the *report_eco_library_cells* command instead.

`-power_attribute` *power_attribute_name*

Specifies a power attribute set on library cells. If you specify this option, the summary of power attribute value for each cell group is reported in a separate column.

Description

This command reports the summary of cell usage in a design.

In a multi-scenario analysis flow, this command must be used at the worker processes.

This command reports the usage of cells based on the specified pattern when the *-pattern_priority* option is used. If the *-pattern_priority* option is not used, the command reports a summary of cell usage for each cell group.

Cells are categorized into the following groups:

- Combinational - combinational cells
- Sequential - sequential cells such as latches and flip-flops
- Clock - cells in the clock network
- Others - other cells, including I/O pad, memory, and black-box cells

In a HyperScale hierarchical analysis flow, cells within HyperScale blocks are excluded when reporting the cell usage, as their cell counts can vary depending on how the Hyperscale model was abstracted. Thus, the cell usage number reported in a HyperScale analysis can be smaller than the number reported in a full-flat analysis.

Examples

The following example reports HVT and LVT cell counts in the design:

```
pt_shell> report_cell_usage -pattern_priority {HVT LVT}

****************************************
Report : cell_usage
        -pattern_priority { HVT LVT }
Design : design
 ...
****************************************

Pattern   Combinational cell     Sequential cell               Total
--------------------------------------------------------------------------
HVT           1515979 ( 32%)         1291 (  0%)      1517270 ( 32%)
LVT           2474715 ( 53%)       696222 ( 15%)      3170937 ( 68%)
Others           4115 (  0%)          532 (  0%)         4647 (  0%)
```

The following example reports summary of cell count and area for each cell group:

```
pt_shell> report_cell_usage
 ...

Cell Group                    Count                    Area
-----------------------------------------------------------
Combinational             5 ( 71%)            33.16 ( 59%)
Sequential                1 ( 14%)            16.23 ( 29%)
Clock                     1 ( 14%)             6.35 ( 11%)
Others                    0 (  0%)             0.00 (  0%)
-----------------------------------------------------------
Total                     7 (100%)            55.74 (100%)
```

The following example reports summary of cell count, area, and power attribute for each cell group:

```
pt_shell> report_cell_usage -power_attribute pwr_attr
 ...

Cell Group            Count                    Area              Power_Attr
-------------------------------------------------------------------------
-------
Combinational             5 ( 71%)            33.16 ( 59%)         1815.87
 ( 66%)
Sequential                1 ( 14%)            16.23 ( 29%)          548.05
 ( 20%)
Clock                     1 ( 14%)             6.35 ( 11%)          390.13
 ( 14%)
Others                    0 (  0%)             0.00 (  0%)            0.00
 (  0%)
-------------------------------------------------------------------------
-------
Total                     7 (100%)            55.74 (100%)         2754.05
 (100%)
```

See Also

- fix_eco_leakage
- fix_eco_power

---

report_check_ctpm_tolerance

Reports the passing threshold used to report passing rate in *gen_ctpm* and *validate_ctpm*.

Syntax

string *report_check_ctpm_tolerance*

Feedback

Description

The *report_check_ctpm_tolerance* command is used to report tolerance used in *gen_ctpm* and *validate_ctpm* to calculate passing rate. The tolerance is setup by *set_check_ctpm_tolerance* command. If abs(error percentatage) <= rel_tol or abs(error_in_ns)<abs_tol, a data point is considered as passing.

Examples

The following script set delay and slew passing tolerance as 5%/5ps, and report the user-defined tolerance for CTPM checking.

```
set ps_enable_analysis true
set ps_enable_spice2design_analysis true

set_check_ctpm_tolerance -delay {0.05 0.005} -slew {0.05 0.005}
pt_shell> report_check_ctpm_tolerance
  type                          relative_error    absolute_error
  delay                           0.050000            0.005000
  slew                            0.050000            0.005000
  constraint_setup                0.100000            0.010000
  constraint_hold                 0.100000            0.010000
  constraint_min_pulse_width      0.100000            0.010000
  delay_sigma                     0.030000            0.003000
  slew_sigma                      0.030000            0.003000
  constraint_setup_sigma          0.100000            0.010000
  constraint_hold_sigma           0.100000            0.010000
  pin_cap                         0.030000            0.000100
1
```

See Also

- gen_ctpm
- set_check_ctpm_tolerance
- validate_ctpm
- ps_enable_spice2design_analysis

---

report_clib

Displays information about timing data of the specified CLIB reference library, which which has been constructed from one or more .db libraries.

Syntax

status *report_clib*

```
[-nosplit]
[-timing_arcs]
```

```
[-power_arcs]
[-verbose]
library
[lib_cell_list]
```

Data Types

```
library          list
lib_cell_list    list
```

Arguments

`-nosplit`

Most of the library information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

`-timing_arcs`

Reports all cell timing arcs from the library.

`-power_arcs`

Reports all cell power arcs from the library.

`-verbose`

Possibly report additional detail.

`library`

Specifies the CLIB reference library to report. This can be either a .db library name or a collection of a single .db library. The CLIB library that contains the specified .db library will be reported.

`lib_cell_list`

Displays a list of library cells about which information is reported. The default is to report information about all cells in the technology library. Each element in this list is either a collection of library cells or a pattern that matches library cells in the *library* option.

Description

The *report_clib* command displays timing information from the specified CLIB reference library, including the PVT parameters and source .db libraries of each pane.

The default timing data includes the power rails, the details for each timing pane, the constituent logic libraries, and the operating conditions. You can also include the cell timing arcs by using the *-timing_arcs* option.

The scaling group and min-lib relationships (if any) of each pane are also reported.

This command exists only in a CLIB-enabled build.

Examples

The following command generates a default report for the my_lib_fast reference library.

```
prompt> report_clib ccs_1.1.db:synop_lib
****************************************
Report : library
Library: synop_lib
Version:
****************************************

Design count: 1

--------------------------------------------------------------------------------
-------
** Timing Data:

Power rails:

Index  Name         Type
  0    <default>    power

Pane count: 4

Pane 0:
  Process label: (none)
  Process number: 1
  Voltage rail count: 1
  Voltage for rail 0 (<default>): 1.1
  Temperature: 125

  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
    l/h RiseTrans:  0.3/0.7  h/l FallTrans:   0.7/0.3
    TransDerate:    0.4


  In link path.
  Min_library: pane 1.
  In full scaling lib group with panes 0 3.


Pane 1:
  Process label: (none)
  Process number: 1
  Voltage rail count: 1
  Voltage for rail 0 (<default>): 1.3
  Temperature: 125

  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
```

```
    l/h RiseTrans:  0.3/0.7  h/l FallTrans:   0.7/0.3
    TransDerate:    0.4

  Source .db file:

 /remote/us01home22/dbraun/work/clients/dbraun_dgplt_main/unit_clt/librar
ies/ccs_1.3.db

  Referenced as min_library by 1 pane.
  In full scaling lib group with panes 1 2.


Pane 2:
  Process label: (none)
  Process number: 1
  Voltage rail count: 1
  Voltage for rail 0 (<default>): 0.9
  Temperature: 125

  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
    l/h RiseTrans:  0.3/0.7  h/l FallTrans:   0.7/0.3
    TransDerate:    0.4

  Source .db file:

 /remote/us01home22/dbraun/work/clients/dbraun_dgplt_main/unit_clt/librar
ies/ccs_0.9.db

  No min_library.
  In full scaling lib group with panes 1 2.


Pane 3:
  Process label: (none)
  Process number: 1
  Voltage rail count: 1
  Voltage for rail 0 (<default>): 1.3
  Temperature: 125

  Thresholds:
    r/f InputDelay: 0.5/0.5  r/f OutputDelay: 0.5/0.5
    l/h RiseTrans:  0.3/0.7  h/l FallTrans:   0.7/0.3
    TransDerate:    0.4

  Source .db file:

 /remote/us01home22/dbraun/work/clients/dbraun_dgplt_main/unit_clt/librar
ies/ccs_1.2.db

  No min_library.
  In full scaling lib group with panes 0 3.
```

```
Data Model Existence :
  nldm


Source .db libraries:
  ccs_1.1.db:synop_lib
  ccs_1.3.db:synop_lib
  ccs_0.9.db:synop_lib
  ccs_1.2.db:synop_lib


Operating Conditions:

   Name                  Process   Label                Temp      Voltage
  Original DB Name     Original DB Filename

 ------------------------------------------------------------------------
----------------------------------------------------------------
   nom_pvt               1.00      (none)               125.00    1.10
  synop_lib            ccs_1.1.db
   nom_pvt               1.00      (none)               125.00    1.30
  synop_lib            ccs_1.3.db
   nom_pvt               1.00      (none)               125.00    0.90
  synop_lib            ccs_0.9.db
   nom_pvt               1.00      (none)               125.00    1.30
  synop_lib            ccs_1.2.db



1
```

See Also

- get_libs

---

report_clock

Reports clock-related information.

Syntax

status *report_clock*

```
[-attributes]
[-skew]
[-groups]
[-map]
[-skip_internal]
[-map_of instance_list]
```

```
[-cells hierarchical_cell_list]
[-include internal | virtual]
[-exclusivity]
[-nosplit]
[-sms_scenarios sms_scenarios_list]
[clock_names]
```

Data Types

```
clock_names                 list
sms_scenarios_list          collection
hierarchical_cell_list      list
instance_list               list
```

Arguments

`-attributes`

Reports the clocks in the design, along with additional information such as source type, signal rise and fall times, and attributes. This report is shown by default.

`-skew`

Reports clock latency (source and network latency) and uncertainty information set on the design by the *set_clock_latency* and *set_clock_uncertainty* commands, respectively.

The latency reported by this option includes propagated latency for generated clocks.

Clock network latency information includes rise latency and fall latency. Clock source latency information includes rise latency and fall latency for early and late arrivals. Clock uncertainty information includes intraclock setup or hold uncertainty and interclock setup or hold uncertainty. This option also reports any fixed clock transitions set by the *set_clock_transition* command. This option reports only active clocks.

`-groups`

Reports the clock groups defined for the design, including the list of active clocks in the current analysis scope and grouping of exclusive clocks and asynchronous clocks defined by using the *set_clock_groups* command.

To query specific clocks or clock relationships, use the *get_clock_relationship* command.

`-map`

Shows the map between top-level clocks and block-level clocks in HyperScale.

`-skip_internal`

Used together with the *-map* option, skips internal clocks in the mapping between top-level clocks and block-level clocks in HyperScale.

`-map_of` *`instance_list`*

Shows the map between top-level clocks and block-level clocks in the specified HyperScale instances. If the *-map_of* option is not used, the *-map* option reports across all HyperScale instances.

`-cells` *`hierarchical_cell_list`*

Shows clocks that physically enter the specified hierarchical cells through certain hierarchical pins of the cells.

`-include internal | virtual`

Used together with the *-cells* option, optionally also shows the internal clocks of the hierarchical cell and virtual clocks that do not physically enter the cell but do launch/capture data to/from the cell. Valid values are *internal* and *virtual*. If *-include internal* is used, the Pins column shows the source pins of the internal clock. If *-include virtual* is used, the Pins column shows the relevant data pins of the virtual clock; the 'Clock type' column shows 'v' to indicate virtual clocks.

`-nosplit`

Specifies not to split lines if a column overflows. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

`-exclusivity`

Reports the clock exclusivity points in the design, whether inferred implicitly from clocks passing through MUX cells (when the *timing_enable_auto_mux_clock_exclusivity* variable is set to *true*) or defined explicitly by the *set_clock_exclusivity* command.

`clock_names`

Specifies the clocks to be reported. The default is all clocks.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only clock information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Feedback

Description

Reports clock-related information. The report contents are controlled by the options used. If you do not specify any options, the command reports all clocks in an *-attributes* report.

When a clock is created, it defaults to being active. Clocks can subsequently be set active/inactive using the *set_active_clocks* command. The default report shows which clocks are active.

If a dynamic component of clock latency has been specified by the *set_clock_latency* command, the *-skew* option can be used to report what values have been set. In this case, all components of clock latency will be reported.

Some information, such as the waveform of a generated clock or propagated skew, is shown only after a timing update (from an explicit *update_timing* command or an implicit timing update caused by another command). This behavior is intended to help you to report clocks without incurring the cost of a complete timing update.

To show the clock mapping used in HyperScale between top and block, use the *report_clock -map* command. Any user-specified mappings from the *set_clock_map* command are annotated with an asterisk (*) in the report.

To obtain a list of all clocks in the design, use the *all_clocks* command.

Examples

Here are some examples of the *report_clock* command:

```
pt_shell> create_clock -period 20.000000 [get_ports {CLK}]
pt_shell> report_clock
****************************************
Report : clock
Design : counter
 ...
****************************************

Attributes:
    p - propagated_clock
    G - Generated clock

Clock          Period   Waveform            Attrs     Sources
------------------------------------------------------------------------
CLK             20.00   {0 10}                        {CLK}

pt_shell> set_clock_latency 2.4 [get_clocks CLK]
pt_shell> set_clock_latency 0.17 -source -early [get_clocks CLK]
pt_shell> set_clock_latency 0.19 -source -late  [get_clocks CLK]
pt_shell> set_clock_uncertainty 1.3 [get_clocks CLK]
pt_shell> set_clock_transition  1.7 [get_clocks CLK]
pt_shell> report_clock -skew
****************************************
```

```
Report : clock_skew
Design : counter
 ...
****************************************

                  Rise           Fall           Hold           Setup
Object           Delay          Delay        Uncertainty    Uncertainty
-------------------------------------------------------------------------
CLK               2.40           2.40           1.3            1.3

-------------------------------------------------------------------------
                                 Source Latency
-------------------------------------------------------------------------
Object         Min Rise       Min Fall       Max Rise       Max Fall
-------------------------------------------------------------------------
CLK               0.17           0.17           0.19           0.19

                  Rise           Fall
Object           Transition     Transition
-------------------------------------------------------------------------
CLK               1.7            1.7

pt_shell> set_clock_latency -late -source -dynamic 0.5 4.5 [get_clocks
 clk]
pt_shell> set_clock_latency -early -source 2.5 -dynamic 0.5 [get_clocks
 clk]
pt_shell> report_clock -skew
****************************************
Report : clock_skew
Design : latency
 ...
****************************************

                Min Condition Source Latency      Max Condition Source
 Latency
-------------------------------------------------------------------------
-----------------
Object         Early_r Early_f  Late_r  Late_f   Early_r Early_f  Late_r
 Late_f  Rel_clk
-------------------------------------------------------------------------
-----------------
clk (static)      2.00    2.00    4.00    4.00      2.00    2.00    4.00
  4.00

         --
clk (dynamic)     0.50    0.50    0.50    0.50      0.50    0.50    0.50
  0.50

         --

 -------------------------------------------------------------------------
---
```

```
clk (total)      2.50    2.50    4.50    4.50      2.50    2.50    4.50
  4.50

pt_shell> set_clock_groups -ex -group {clk1 clk3}
pt_shell> set_clock_groups -asyn -name a1 -group clk2 -group clk2
pt_shell> set_active_clocks {clk1 clk2}
pt_shell> report_clock -groups
****************************************
Report : clock_groups
 ...
****************************************

Active clocks:
  clk1  clk2

Total exclusive groups: 1.
NAME : clk1_others
         -group {clk1 clk3 }
         -group {clk2 clk4 }

Total asynchronous groups: 1.
NAME : a1
         -group {clk2 }
         -group {clk4 }

pt_shell> report_clock -map
****************************************
Report : clock
Design : BLOCK_C
 ...
****************************************

Instance              Top level clock     Block level clock
-------------------------------------------------------------
design                SYSCLK1             BLK_SYSCLK

pt_shell> report_clock -cell [get_cell {core1 core2}] -include {virtual
 internal}
****************************************
Report : clock of hierarchical cell
Design : wrapper
 ...
****************************************

Clock type:
    v - Virtual clock

Cell                    Clock               Clock type     Pins
-------------------------------------------------------------------------
-------
core1                   A                                  core1/clk1
                        B                                  core1/clk1
                        X                   v              core1/out2
```

```
                          n1_t4_CLK           v               core1/clk2
                          n_t2c4_CLK          v               core1/in
                          C                   v               core1/out1
core2                     X                                   core2/clk1
                          n1_t4_CLK           v               core2/clk2
                          n_t2c4_CLK          v               core2/out1
 core2/out2
                          C                   v               core2/in

pt_shell> report_clock -cell [get_cell {U_BLOCK}] -include {virtual
 internal}
****************************************
Report : clock of hierarchical cell
Design : top
 ...
****************************************

Clock type:
    v - Virtual clock

Cell                      Clock               Clock type      Pins
--------------------------------------------------------------------------------
-------
U_BLOCK                   top_CLK_BOUND                       U_BLOCK/side
                          BLK_gCLK                            U_BLOCK/in
 U_BLOCK/U4/Z
                          top_CLK_BOUND       v               U_BLOCK/out2
                          BLK_gCLK            v               U_BLOCK/clk

pt_shell> report_clock -exclusivity
****************************************
Report : clock
 ...
****************************************

clock exclusivity points:

    -type       mux
    -output     MUX/Y
```

See Also

- all_clocks
- create_clock
- get_clock_relationship
- remove_clock_groups
- report_transitive_fanout
- set_active_clocks

Feedback

- set_clock_groups
- set_clock_latency
- set_clock_map
- set_clock_transition
- set_clock_uncertainty
- set_clock_exclusivity
- timing_enable_auto_mux_clock_exclusivity

---

report_clock_crossing

Generates a report about paths launched from one clock and captured by another.

Syntax

string *report_clock_crossing*

```
[-from from_list]
[-to to_list]
[-include include_list]
[-format format]
[-output file_name]
[-nosplit]
[-verbose]
[-ignore_invalid]
[-period]
[-edge_relationships]
```

Data Types

```
from_list     list
to_list       list
include_list  list
```

Arguments

`-from from_list`

Specifies a list of clocks to report clock crossing from. If no *-from* argument is given, all from clocks are reported.

`-to to_list`

Specifies a list of clocks to report clock crossing to. If no *-to* argument is given, all to clocks are reported.

Feedback

-include *include_list*

Specifies a list of type of crossing to report. By default, all types are reported.

" valid - If some of the paths are *false* due to the *set_false_path* command, the interaction is still considered valid, but is marked with a '#'. " false - All of the paths are *false* due to the *set_false_path* command. The interaction is marked with an 'F'. " physically_exclusive - All of the paths are *false* due to a physically exclusive clock grouping. The interaction is marked with a 'P'. " logically_exclusive - All of the paths are *false* due to a logically exclusive clock grouping. The interaction is marked with an 'L'. " asynchronous - All of the paths are *false* due to an asynchronous clock grouping. The interaction is marked with an 'A'.

Allowed values are as follows:

• *valid*:
Reports clock interactions where there are valid paths.
If some of the paths are **false**, that is noted with an attribute
of '#'.

• *false*:
Reports clock interactions where all of the paths between the
clocks are
declared as **false**. Only clock interactions that are connected
with paths
that are declared as **false** are reported. If there is a clock
interaction
declared as **false** but the design has no paths connecting those
two clocks, that
interaction is not reported.

• *logically_exclusive*:
Reports clock interactions where all the paths between the clocks
are
declared as logically_exclusive with a **set_clock_groups** command.
Only clock interactions that are connected with paths are
reported.

• *physically_exclusive*:
Reports clock interactions where all the paths between the clocks
are
declared as physically_exclusive with a **set_clock_groups** command.
Only clock interactions that are connected with paths are
reported.

• *asynchronous*:
Reports clock interactions where all the paths between the clocks
are
declared as asynchronous with a **set_clock_groups** command.

```
   Only clock interactions that are connected with paths are
    reported.
```

`-format` *`format`*

Specifies the format of the report. The default for format must be either standard or csv. The default is standard. The default generates a text report. If csv is specified, the report is written out in a comma-separated value format. If csv is specified, the *-verbose* option is implied. This option requires the *-output* option.

`-output` *`file_name`*

Specifies the output file name for the output.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing tools to extract information from the report output.

`-verbose`

Adds more things to the report. One main addition is all pairs of clocks are added with "No paths" for the noninteracting clocks. In addition, the clock waveforms are also printed and minimum edge separation is printed separately when *-edge_relationships* is used.

`-ignore_invalid`

Reports clock group relation between the launch and capture clock pair even if no valid timing path exists between them.

`-period`

Adds the periods of each clock after each clock name and if there is a valid crossing the base period for the clocks is displayed. If there are no valid paths between the clocks, the base period is given as "--". If the base period is more than 101 times the smallest period, the base period is marked with "*". See the following example for more information.

`-edge_relationships`

Shows the most constraining edge relationship between the clocks that have valid crossing along with multicycle path information. This option also forces the *-period* option to be used so that the periods are displayed.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Generates a report showing the clocks that launch datapaths that are capture by a different class. The from clock column gives the launching clock and the attribute column

is blank for only valid paths. The following attributes are given to classify the interactions between the clocks in the row:

```
#     - some paths false
    F     - all paths false
    L     - logically exclusive clock group
    P     - Physically Exclusive Clock Groups
    A     - Asynchronous Clock Groups
    A_MCP - all multicycle paths
    S_MCP - some multicycle paths
    A_MMD - all min-max delay paths
    S_MMD - some min-max delay paths
    MG    - from clock master of to clock
    GM    - to clock master of from clock
    N     - No valid path between the clock pair ( when -verbose option
 added )
    V     - Valid path between the clock pair    ( when -verbose option
 added )
    *     - Base Period Limit Exceeded ( when -period option is
 specified )
```

The Crossing Clocks column gives all of the clocks that capture paths from the "from clock" with the same attribute. The crossing clocks are separated by spaces and are split onto the next line unless the *-nosplit* option is given.

In addition to the default output, the *-format* and the *-output* options can be used to write the clock crossing report to a file in comma-separated value format.

Examples

The following example shows the summary report.

```
pt_shell> report_clock_crossing -nosplit
****************************************
Report : report_clock_crossing
Design : test
Scenario: default
Version: ...
Date   : ...
****************************************

Attributes
    P     - physically exclusive clock groups
    L     - logically exclusive clock groups
    A     - asynchronous clock groups
    F     - all paths false
    #     - some paths false
    A_MCP - all multicycle paths
    S_MCP - some multicycle paths
    A_MMD - all min-max delay paths
    S_MMD - some min-max delay paths
    MG    - from clock master of to clock
```

```
    GM    - to clock master of from clock

From Clock           Attr     Crossing Clocks
-------------------------------------------------------------------------
-------
CLK                  #,MG     GCLK
GCLK                          CLK2
GCLK                 A_MCP,GM CLK
```

The following example shows reporting the crossings from clock GCLK.

```
pt_shell> report_clock_crossing -from GCLK -nosplit
****************************************
Report : report_clock_crossing
Design : test
Scenario: default
Version: ...
Date   : ...
****************************************

Attributes
    P     - physically exclusive clock groups
    L     - logically exclusive clock groups
    A     - asynchronous clock groups
    F     - all paths false
    #     - some paths false
    A_MCP - all multicycle paths
    S_MCP - some multicycle paths
    A_MMD - all min-max delay paths
    S_MMD - some min-max delay paths
    MG    - from clock master of to clock
    GM    - to clock master of from clock

From Clock           Attr      Crossing Clocks
-------------------------------------------------------------------------
-------
GCLK                           CLK2
GCLK                 A_MCP,GM  CLK
```

The following example shows using the *-period* option. Note that the base period of "--" is given if there are no valid paths. The base period is marked with "*" if it exceeds 101 times the smallest period. Also, note that this option forces only two clocks to be reported per line.

```
pt_shell> report_clock_crossing -period -nosplit
****************************************
Report : report_clock_crossing
Design : test1
Scenario: default
Version: ...
Date   : ...
****************************************
```

```
Attributes
    P     - physically exclusive clock groups
    L     - logically exclusive clock groups
    A     - asynchronous clock groups
    F     - all paths false
    #     - some paths false
    A_MCP - all multicycle paths
    S_MCP - some multicycle paths
    A_MMD - all min-max delay paths
    S_MMD - some min-max delay paths
    MG    - from clock master of to clock
    GM    - to clock master of from clock
    *     - Base Period Limit Exceeded

Base
Period  From Clock           Attr   Crossing Clocks
-------------------------------------------------------------------------
-------
10.64   CLK(5.32)            MG     GCLK(10.64)
1143.32* CLK3(11.32)         MG     GCLK(10.64)
--      GCLK(10.64)          F      CLK2(15.00)
10.64   GCLK(10.64)          GM     CLK(5.32)
1143.32* GCLK(10.64)         GM     CLK3(11.32)
```

The following example shows using the *-edge_relationships* option. Note that for clocks that have valid clock crossing it shows the most constraining edge relationship.

```
pt_shell> report_clock_crossing -edge_relationships -nosplit
****************************************
Report : report_clock_crossing
Design : test
Scenario: default
Version: ...
Date   : ...
****************************************

Attributes
    P     - physically exclusive clock groups
    L     - logically exclusive clock groups
    A     - asynchronous clock groups
    F     - all paths false
    #     - some paths false
    A_MCP - all multicycle paths
    S_MCP - some multicycle paths
    A_MMD - all min-max delay paths
    S_MMD - some min-max delay paths
    MG    - from clock master of to clock
    GM    - to clock master of from clock
    *     - Base Period Limit Exceeded

Base
Period  From Clock   Attr      Crossing Clocks   Setup edge Hold edge
```

```
------------------------------------------------------------------------
---
20.00    CLK(10.00)    #,MG       GCLK(20.00)        R(--)->R(--)[MCP 1]
                                                     R(0.00)->R(0.00)[MCP 0]
--       GCLK(20.00)   F          CLK2(15.00)
20.00    GCLK(20.00)   A_MCP,GM   CLK(10.00)         R(0.00)->R(30.00)[MCP 3]
                                                     R(0.00)->R(20.00)[MCP 0]
20.00    GCLK(20.00)   A_MCP,GM   CLK(10.00)         F(10.00)->R(40.00)[MCP
 3]
                                                     F(10.00)->R(30.00)[MCP
 0]
```

See Also

- get_clock_crossing_points

---

report_clock_gate_savings

Reports toggle savings on clock gates.

Syntax

string *report_clock_gate_savings*

```
[-by_clock_gate]
[-sequential]
[-hierarchical]
[-nosplit]
[-clocks clock_list]
[-sort_by sort_method]
[object_list]
```

Data Types

clock_list	list
sort_method	string
object_list	list

Arguments

`-by_clock_gate`

With this option two tables are generated. One is a summary table with column of clock name, total registers, stages, number of clock gate per stage, number of gated cells (register + clock gates) and Clock Gating Efficiency of that stage. The other is detailed clock gating structure table which contains clock name, clock gate stage, gating element, fanout, number of registers, number of gate cells, Toggle Savings, Clock Gating Efficiency, List of gate cells.

`-sequential`

Specifies that the report is to report on registers in the design. When the *-sequential* option is applied, the command uses toggle rates to determine the fraction of clock events at registers that result in a toggle on the Q pin. This fraction is a measure of how well clock gated a register is. Values near 25% are near optimal for datapath registers. Lower values indicate there might be more room for clock gating, as the clock is toggling often when the Q pin toggles little.

In addition to the Q/Clk toggle ratio, the Register Gating Efficiency is reported for each register. This measures the ratio between the toggle rate of the local (gated) clock, and the root clock specified with create_clock. High values indicate that most of the toggles from the root clock have been suppressed by clock gates along the clock tree from the root to the register.

When the *-by_clock_gate* and *-sequential* options are used together, the command reports on register banks in the design. Registers are grouped into clusters by the clock gates that drive their clock pins. An additional cluster of ungated registers is also reported included in the report.

`-hierarchical`

Specifies that results are to be aggregated into hierarchical blocks, and each block is to be reported. Without this option, individual clock gates or registers are reported. The data is aggregated by averaging toggle savings over clock gates in a module.

The *-hierarchical* option cannot be used when both the *-by_clock_gate* and *-sequential* options are also used together.

`-nosplit`

Specifies that report lines should not be split when names are longer than the space provided in the report. This option can make it easier for scripts to parse the results, but might make it harder for humans to read the columns.

`-clocks` *`clock_list`*

Specifies that objects not in the clock domain of one of the given clocks should be excluded from the report.

`-sort_by` *`sort_method`*

Specifies the order in which to print design objects. You can specify the *-sort* option with the *-sequential* and *-by_clock_gate* options but not with the *-hierarchical* option. You can specify the following sort methods:

```
-by_clock_gate              -sequential
----------------            --------------
name                        name
clock_gating_stage          clock_gating_stage
```

```
clock_toggle_rate           clock_toggle_rate
toggle_savings              q_clk_ratio
clock_gating_efficiency     q_toggle_rate
                            gating_efficiency
```

object_list

Specifies that the given cells are to be reported on only. If cells in the list are hierarchical, all clock gates (or registers, if the *-sequential* option is used) under the hierarchical cells are reported on. If the *-sequential* option is used without the *-by_clock_gate* option, any leaf cells in the list should be registers. If the *-by_clock_gate* option is used, or if both the *-by_clock_gate* and *-sequential* options are used, any leaf cells should be individual clock gates.

In the default report, the *object_list* option can also contain clocks, in which case only those clocks are reported on.

Description

This command uses toggle rates in the design to report on the toggle savings on the clock tree from clock gating in the design. To use the command, you should first apply switching activities before the command. In the time based mode, this can be done using the *read_vcd* command followed by the *update_power* command.

In the averaged power mode, apply switching activities by using the *read_saif*, *set_switching_activity*, or *read_vcd* command. In the averaged mode, partial switching activity annotation causes the tool to propagate switching activities.

The usefulness of the *report_clock_gate_savings* command is largely dependent on the quality of the switching activities. To be useful, the activities should come from simulation. The testbench used from simulation should be designed to simulate the typical behavior of the design and should cover all modes that are part of the typical behavior. If a mode is left off, some part of the design might not be exercised.

The default report displays overview information about each clock and the registers that the clock drives. You can use this report to summarize the overall effectiveness of clock gating in the design. The report computes the average toggle savings over the registers driven by each clock, where here the toggle savings for a register is defined as the ratio of the toggle rate at the clock pin of the register to the toggle rate of the clock. A 20% toggle savings means that the toggle rate at clock pin of a register is 80% of the toggle rate of the main clock for the clock domain. When a register is in several clock domains, the register is only included with the domain of the faster clock, so that registers are not included twice in the report.

The *-sequential* reports helps to identify registers with many clk events but few Q toggles.

When the Q/Clk ratio for registers is significantly lower than 25%, there can be room to improve the clock gating for these registers by adding clock gating or increasing the

fraction of time that clock gating suppresses clock toggles. A brief explanation of why 25% can be regarded as optimal is below.

In the case of a datapath, during cycles when the datapath is exercised, the clock gating is enabled, and the probability of Q toggling in a cycle is approximately 50%, assuming random and independent data is moving through the datapath. On cycles when the datapath is not being exercised, the clock gating should be disabled. In this idealized situation, the Q/clk toggle ratio should be 25%, since the clk toggles twice in a cycle, and the Q toggles on average one time every other cycle (a 1:4 ratio).

There are some exceptions where registers can have Q/Clk ratios larger than 25%, such as low bits of counters where the Q should toggle every cycle when the counter is active. However, on the whole, most registers in the design should report (using the *-sequential* option) a Q/Clk ratio less than 25%.

More information can be gathered about individual clock gates and their associated registers when the *-by_clock_gate* and *-sequential* options are used together. In this case, Individual clock gates are reported, with each clock gate followed by a list of the registers driven by the clock gate. This report can be more useful than the *-sequential* report alone in helping find register banks with low Q/Clk ratios, where it is possible to disable the clock gate more often. The report can also help find register banks that should be split; it might be that a portion of the registers driven by a clock gate have low Q/Clk ratios, and these registers might be able to have their own separate clock gate, which could allow disabling the clock more often for those registers.

The *-hierarchical* reports (which can be used with either the *-by_clock_gate* or *-sequential* option) can help to identify modules of interest. Since many register banks are too small for clock gating, the hierarchical sequential report might be less useful as the register banks of interest are overwhelmed in the aggregated data by the small register banks which are not clock gated.

USAGE

You can use the default report to view the overall effectiveness of clock gating in the design.

Examples

The following example shows the default report:

```
report_clock_gate_savings
****************************************
Report : Clock Gate Savings
         power_mode: Averaged
Design : top
****************************************

--------------------------------------------------------------------------------
-----------
```

```
Clock: clk
  + Clock Toggle Rate: 0.2
  + Number of Registers: 200
  + Number of Clock Gates: 29
  + Average Register Gating Efficiency (savings with respect to root
 clock): 48.1%
--------------------------------------------------------------------------
-----------
Toggle Savings                          Number of       % of
Distribution                            Registers       Registers
--------------------------------------------------------------------------
-----------
100%                                    8               4.0%
80% - 100%                              13              6.5%
60% - 80%                               71              35.5%
40% - 60%                               71              35.5%
20% - 40%                               11              5.5%
0%                                      24              12.0%
--------------------------------------------------------------------------
-----------
```

The following example shows the *-by_clock_gate* report on a particular module. There are two individual clock gates, with toggle rates shown.

```
pt_shell> report_clock_gate_savings -by_clock_gate

****************************************
Report : Clock Gate Savings
        power_mode: Averaged
        -by_clock_gate
 ...
****************************************

--------------------------------------------------------------------------
---------
                          Clock Gate Structure Summary
--------------------------------------------------------------------------
---------
Clock     Total         Clock Gate     # of Clock     # of
 Clock Gating
          Registers     Stage          Gates          Registers
 Efficiency
--------------------------------------------------------------------------
---------
CLK       12            0              0              0              NA
                        1              2              8
 30.61%
                        2              1              4
 54.52%
--------------------------------------------------------------------------
---------
```

```
------------------------------------------------------------------------
--------------------
                                      Clock Gate Structure Details
------------------------------------------------------------------------
--------------------
Clock  Clock  Gating                # of   # of    # of       Toggle
 Clock       Receiver
       Gate   Element               Fan    Clock   Registers  Savings
 Gating      Clock
       Stage                       Out    Gates              (%)
 Efficiency  Gates
------------------------------------------------------------------------
--------------------
CLK    1      clk_gate_R1_reg/latch  4     0       4          81.8%
 27.3%
       1      clk_gate_Q_reg/latch   4     0       4          55.2%
 3.4%
       2      clk_gate_R2_reg/latch  5     1       4          81.8%
 54.5% clk_gate_Q_reg/latch
------------------------------------------------------------------------
--------------------

1
```

The following example shows the hierarchical report for clock gates.

```
pt_shell> report_clock_gate_savings -by_clock_gate -hierarchical

****************************************
Report : Clock Gate Savings
Design : mydesign
         power_mode: Averaged
         -hierarchical
 ...
****************************************
-------------------------------------------
Module              Number         Toggle
                 Clock Gates     Savings (%)
-------------------------------------------
TOP                   225           67%
/A                    125           50%
//B                    30           12%
/C                    100           80%
-------------------------------------------
1
```

The following report shows individual registers (not register banks). The clk/q ratio of 4 is expected for a register clocked every cycle, with random data on the d pin of the register.

```
pt_shell> report_clock_gate_savings -sequential

****************************************
Report : Clock Gate Savings
```

```
Design : mydesign
         power_mode: Averaged
         -sequential
 ...
****************************************
--------------------------------------------------------------------------
------
Register      Root Clock      Local Clock    Q           Q/Clk
 Register
              Toggle          Toggle         Toggle      Toggle
 Gating
              Rate            Rate           Rate        Ratio
 Efficiency
--------------------------------------------------------------------------
------
TOP/A/U34     0.2             0.1          0.025         25%          50%
TOP/A/U35     0.2             0.1          0.01          10%          50%
--------------------------------------------------------------------------
------
1
```

The following example shows the sequential hierarchical report, which averages over registers in each hierarchical block.

```
pt_shell> report_clock_gate_savings -sequential -hierarchical

****************************************
Report : Clock Gate Savings
Design : mydesign
         power_mode: Averaged
         -sequential
         -hierarchical
 ...
****************************************
-------------------------------------------------------------
Module     Number of          Q/Clk              Avg Register
           Registers          Toggle             Gating
                              Ratio              Efficiency
-------------------------------------------------------------
TOP        2250               17.5%              40.6%
 A         1250               15.6%              37%
 B          300               23.3%              70%
 C         1000               15.6%              45%
-------------------------------------------------------------
1
```

See Also

- read_saif
- read_vcd

Feedback

- set_switching_activity
- update_power

---

report_clock_gating_check

Displays clock-gating check information about specified pins.

Syntax

status *report_clock_gating_check*

```
[-significant_digits digits]
[-nosplit]
[-auto_inferred_only]
[-sms_scenarios sms_scenarios_list]
[object_list]
```

Data Types

digits	integer
object_list	list
sms_scenarios_list	collection

Arguments

`-significant_digits digits`

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default value is 2. Use this option if you want to override the default.

`-nosplit`

Prevents line splitting and facilitates scripting to extract information from the report output.

`-auto_inferred_only`

Shows only the automatically inferred clock-gating checks.

object_list

Specifies a list of cells, pins, clocks, or design objects to report.

`-sms_scenarios sms_scenarios_list`

Specifies a collection of SMS scenarios to analyze. Only clock-gating checks compatible to the specified SMS scenarios collection are reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Feedback

Description

The *report_clock_gating_check* command displays the following information for the clock-gating checks. Information displayed about the specified objects: Instance of the cell, enable pin, clock pin, setup/hold time for rise, setup/hold time for fall, user-specified high/low active waveform. The asterisk (*) in the high/low field means that the user-specified high/low overrides what PrimeTime inferred from the logic of the cell.

The following attributes show where the clock-gating check originally was defined:

- I - Automatically inferred by PrimeTime.
- P - Power Compiler inserted the check.
- L - Defined by library cell.

Examples

The following example displays how to report all clock-gating check informations.

```
pt_shell> report_clock_gating_check
****************************************
Report : clock-gating check
Design : flop123
 ...
****************************************

                                    Rise                Fall
Cell      Enable  Clock     Setup     Hold      Setup     Hold    High/Low
 Attr
--------------------------------------------------------------------------
------
mid       A       B         5.00      3.00      5.00      3.00    High
 I
MX        A       S         5.00      3.00      5.00      3.00    High
 I
MX        B       S         5.00      3.00      5.00      3.00    Low (*)
 I

Note: * indicates user override of tool inferred controlling value
Attr: I:auto inferred, P:power compiler inserted, L:library cell defined
```

See Also

- report_constraint
- report_timing
- set_clock_gating_check
- report_default_significant_digits

Feedback

report_clock_jitter

Reports clock jitter information previously set by the *set_clock_jitter* command.

Syntax

```
status report_clock_jitter
   [-clock clock_list]
```

Data Types

```
clock_list     list
```

Arguments

-clock *clock_list*

Specifies a list of clocks to report. If you do not use this option, the command reports the clock jitter for all the clocks whose master (or itself) has cycle-to-cycle jitter or duty-cycle jitter.

Description

This command reports the cycle-to-cycle jitter and the duty-cycle jitter of the clock. If the clock is a generated clock, it will report the cycle-to-cycle and the duty-cycle jitter of its primary master clock. For example, if the generated clock G2 is created from the generated clock G1 and the generated clock G1 is created from the primary master clock M, the cycle-to-cycle jitter and the duty-cycle jitter reported for G2 is in fact the cycle-to-cycle jitter and the duty-cycle jitter of its primary master clock, M. The command reports the primary master clock of the generated clock as well. In case the clock is primary master clock, it will print the clock itself.

To set the clock jitter use the *set_clock_jitter* command.

Examples

The following example reports the cycle-to-cycle jitter of 0.5 and duty-cycle jitter of 0.7 specified on the primary master clock mclk, and its generated clock gclk.

```
pt_shell> set_clock_jitter -clock [get_clocks mclk] -cycle 0.5
 -duty_cycle 0.7
pt_shell> report_clock_jitter

****************************************
Report : clock jitter
 ...
****************************************

Clock         Cycle Jitter   Duty Cycle Jitter   Master Clock with Jitter
--------------------------------------------------------------------------
-------
```

Feedback

```
mclk              0.500          0.700              mclk
gclk              0.500          0.700              mclk
```

See Also

- remove_clock_jitter
- set_clock_jitter

---

report_clock_timing

Reports timing attributes of clock networks.

Syntax

string *report_clock_timing*

```
-type report_type
[-clock clock_list]
[-from_clock from_clock_list]
[-to_clock to_clock_list]
[-from from_list]
[-to to_list]
[-setup | -hold]
[-launch | -capture]
[-rise | -fall]
[-nworst worst_entries]
[-greater_than lower_limit]
[-lesser_than upper_limit]
[-slack_lesser_than slack_upper_limit]
[-include_uncertainty_in_skew]
[-verbose]
[-significant_digits digits]
[-show_clocks]
[-crosstalk_delta]
[-derate]
[-variation]
[-sort_by sort_attr]
[-clock_crossing]
[-include_clock_gating]
[-nosplit]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-sms_scenarios sms_scenarios_list]
[-include_hierarchical_pins]
[-probe]
```

Data Types

```
report_type            string
clock_list             list
from_clock_list        list
```

```
to_clock_list          list
from_list              list
to_list                list
worst_entries          int
lower_limit            float
upper_limit            float
slack_upper_limit      float
digits                 int
sort_attr              string
pre_command_string     string
post_command_string    string
sms_scenarios_list     collection
```

Arguments

`-type report_type`

Specifies the type of report to be generated. You can specify the following values for *report_type*:

- *transition* - Transition time report.
- *latency* - Latency report.
- *skew* - Skew report. You cannot use the *-launch*, *-capture*, *-rise*, *-fall*, and *-lesser_than* options if you specify a skew report, and you can use the *-include_uncertainty_in_skew* option only in a skew, interclock_skew, or summary report.
- *interclock_skew* - Interclock skew report. You cannot use the *-launch*, *-capture*, *-rise*, *-fall*, and *-lesser_than* options if you specify an interclock skew report, and you can use the *-include_uncertainty_in_skew* option only in a skew, interclock_skew or summary report.
- *summary* - Summary report, which shows the worst instances of transition time, latency and skew over the clock networks or subnetworks of interest. If you specify a summary report, you can use only the *-clock*, *-to_list*, *-from_list*, *-include_uncertainty_in_skew*, *-nosplit*, and *-significant_digits* options.

  For nonsummary reports, report entries are ordered with respect to the specified attribute of interest (transition time, latency, or skew). Note that all skews reported are "local" skews. For an explanation of local skew, see the DESCRIPTION section.

`-clock clock_list`

Uses the specified clock networks in the report. A subreport is typically produced for every clock in the *clock_list*, unless you additionally specify the *to_list*, *from_list*, or both options that has no network intersection with a given clock. In that case, PrimeTime drops these clocks from the *clock_list* and also issues a

warning. By default, if you do not specify *clock_list*, all clocks in the design that have associated clock networks are used in the report.

`-from_clock` *`from_clock_list`*

Uses the specified clock networks as from-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report typically considers every clock in *from_clock_list*, unless you additionally specify the *from_list* option that has no network intersection with a given clock. In that case, PrimeTime drops these clocks from the *from_clock_list* and also issues a warning. By default, if you do not specify the *from_clock_list* option, all clocks in the design that have associated clock networks are used as from-clocks in the report.

`-to_clock` *`to_clock_list`*

Uses the specified clock networks as to-clocks in the current interclock skew report. This option can be used only in an interclock skew report. The report typically considers every clock in *to_clock_list*, unless you additionally specify a *to_list* that has no network intersection with a given clock. In that case, PrimeTime drops these clocks from the *to_clock_list* and also issues a warning. By default, if you do not specify the *to_clock_list* option, all clocks in the design that have associated clock networks are used as to-clocks in the report.

`-from` *`from_list`*

Considers the specified sequential clock network pins or ports as from-pins in the current report. If any named pin is not the clock pin of a sequential device, PrimeTime replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list and a warning message is issued.

`-to` *`to_list`*

Considers the specified sequential clock network pins or ports as to-pins in the current report. If any named pin is not the clock pin of a sequential device, such as a sink pin, PrimeTime replaces that pin with all sequential clock pins in the transitive fanout of the named pin. If there are no sequential clock pins in the pin's transitive fanout, the pin is dropped from the list with a warning message.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and *to_list* options respectively; if the *to_list* option is not specified, by default all sink pins in the given clock networks are used. Thus, specifying the *to_list* option reduces the topological scope of the report. For transition time and latency reports, the *from_list* and *to_list* options are concatenated and treated as a single list; if neither is specified, the *to_list* option is assumed to be populated with all sink pins in the given clock networks.

For skew reports, the from-to skew sense is defined by the pins in the *from_list* and *to_list* options respectively; if the *from_list* option is not specified, by default all sink pins in the given clock networks are used. Thus, specifying the *to_list* option reduces the topological scope of the report. For transition time and latency reports, the *from_list* and *to_list* options are concatenated and treated as a single list; if neither is specified, the *to_list* option is assumed to be populated with all sink pins in the given clock networks.

`-setup`

Uses only the setup data path is to be used in the report. Also, the skews, latencies or transition times reported must correspond to those used by the *report_timing* command in the verification of setup constraints. The *-setup* and *-hold* options are mutually exclusive. If neither option is specified, the *-setup* option is assumed. You cannot use this option in summary reports.

`-hold`

Uses only the hold data path in the report. Also, the skews, latencies, or transition times reported must correspond to those used by the *report_timing* command in the verification of hold constraints. The *-setup* and *-hold* options are mutually exclusive. If neither option is specified, the *-setup* option is assumed. You cannot use this option in summary reports.

`-launch`

Uses only pins that launch data in the report. Also, latencies or transition times reported must correspond to those used by the *report_timing* command for sequential device clock pins that are launching data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and *to_list* arguments. In all other reports, the *-launch* and *-capture* options are mutually exclusive. If neither option is specified, the *-launch* option is assumed. You cannot use this option in summary or skew reports.

`-capture`

Uses only pins that capture data in the report. Also, the latencies or transition times reported must correspond to those used by the *report_timing* command for sequential device clock pins that are capturing data. In skew reports, the role is implicit from the from-to sense indicated by the *from_list* and *to_list* options. In all other reports, the *-launch* and *-capture* options are mutually exclusive. If neither option is specified, the *-launch* option is assumed. You cannot use this option in summary or skew reports.

`-rise`

Specifies that the active transition is a rising edge for sequential device clock pins in the current report. The *-rise* and *-fall* options are mutually exclusive; if neither option is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself.

This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. You cannot use this option in summary or skew reports.

`-fall`

Specifies that the active transition is a falling edge for sequential device clock pins in the current report. *-rise* and *-fall* are mutually exclusive; if neither is specified, the active transition at a latch or flip-flop is deduced from the launch or capture role and the behavior of the sequential device itself. This option enables you to answer "what if" questions regarding latency and transition time at sequential device clock pins. You cannot use this option in summary or skew reports.

`-nworst` *`worst_entries`*

Specifies the number of worst report entries to be reported per clock domain; the default is 1. Entries are sorted with respect to the attribute of interest (transition time, latency or skew) specified with the *-type* option of the *report_type* command.

The worst entries are those most likely to cause a violation. For example, if you request a latency report using *-setup* and *-capture*, the smallest *worst_entries* are listed, sorted in ascending order, because small capture latencies for setup paths are more likely to lead to a violation than large capture latencies. By contrast, for skew reports, the worst entries always correspond to the largest skews over the specified domain. You cannot use this option in summary reports.

`-greater_than` *`lower_limit`*

Shows only those entries with an attribute value (latency, transition time or skew) greater (more positive) than *lower_limit*. You cannot use this option in summary reports.

`-lesser_than` *`upper_limit`*

Shows only those entries with an attribute value (latency, transition time or skew) less (more negative) than *upper_limit*. You cannot use this option in summary or skew reports.

`-slack_lesser_than` *`slack_upper_limit`*

Shows only those entries with a slack value less (more negative) than *slack_upper_limit*. For skew report entries slack is the worst slack over all paths launched by the from-pin and captured by the to-pin. For transition time or latency report entries, slack is defined as the worst slack of all paths either launched by a transition at the sink pin (if the *-launch* option is used) or captured by a transition at the sink pin (if the *-capture* option is used).

Feedback

Note that the use of this filter might greatly increase the runtime of this report. However, when used with the *-capture* option the effect on runtime should be small, since PrimeTime can make use of cached slack values to avoid expensive recomputations. You cannot use this option in summary reports.

`-include_uncertainty_in_skew`

Specifies that the user-defined uncertainty between the sequential devices in a launch/capture pair is to be considered a component of skew. You cannot use this option in summary or skew reports.

`-verbose`

Generates a more detailed report; instead of a single line per pin, verbose reports trace the entire source-to-sink path through a clock network to show how the final reported attribute (skew, latency or transition time) was accumulated over the course of the path. You cannot use this option in summary reports.

`-significant_digits` *`digits`*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which has a default of 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

`-show_clocks`

Shows the launching and capturing clocks for every interclock skew entry in the report. This is useful if *from_clock_list* or *to_clock_list* contain more than one clock each. You cannot use this option in interclock skew reports.

`-crosstalk_delta`

Shows the delta delay and delta transition time in verbose reports. The deltas are computed during crosstalk signal integrity analysis, or they can be annotated manually using the *set_annotated_delay -delta_only* and *set_annotated_transition -delta_only* commands. Note that the *-crosstalk_delta* option reports only the calculated or annotated deltas, it does not initiate crosstalk analysis. Only deltas on input pins are shown.

`-derate`

Shows the derating factors in the report. By default, derating factors are not displayed. Derating factors are only shown when you specify the *-verbose* option.

`-variation`

Shows variation information. Using this option is allowed only when detailed variation-aware clock analysis is enabled. Without *-variation*, only the quantile or mean value is displayed, as specified using the *variation_derived_scalar_attribute_mode* variable. With *-variation*, the quantile, mean and either the standard deviation (for skew reports) or sensitivity (for latency or transition reports) is displayed for both verbose and non-verbose reports. In verbose reports, additional statistical info regarding individual path increments appear in new columns, namely the sensitivity, mean and increment, where the increment refers either to the quantile or the effective delay depending on the *variation_report_timing_increment_format* variable.

`-sort_by` *`sort_attr`*

Sorts the entries in the report with respect to the specified variational attribute. This option can be used only with the *-summary* option and only when detailed variation-aware clock analysis is enabled. Note that this option only controls the ranking of entries and not their display. You can specify the following values for *sort_attr*:

- *quantile* - Ranks entries using their quantile values.
- *mean* - Ranks entries using their mean values.
- *sensitivity* - Ranks entries using their sensitivity values. This option cannot be used in a *skew* or *interclock_skew* report.

`-clock_crossing`

Indicates that only skews between interacting clock domains (i.e. clocks connected by at least one non-false data path) is included in the report. This option can only be used in interclock_skew reports.

`-include_clock_gating`

Include clock pins of clock-gating checks in the report. By default, only sequential devices that exercise a setup/hold check are reported by report_clock_timing. This option additionally allows clock-gating checks to be reported.

`-nosplit`

Prevents line-splitting and facilitates writing software to extract information from the report output. If this option is not used, most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`-pre_commands` *`pre_command_string`*

This option is available only if you invoke PrimeTime with the *-multi_scenario* option. This option allows you to specify a string of commands to be executed in the worker context before the execution of the merged reporting command. Commands must be grouped using the ";" character. The maximum size of a command is 1000 chars.

`-post_commands` *`post_command_string`*

This option is available only if you invoke PrimeTime with the *-multi_scenario* option. This option allows you to specify a string of commands to be executed in the worker context after the execution of the merged reporting commands. Commands are grouped using the ";" character. The maximum size of a command is 1000 chars.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only clock timing compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

`-include_hierarchical_pins`

Includes hierarchical pins in the timing report. This option is valid only when -verbose option is set to TRUE.

`-probe`

Specifies that latency reporting should occur directly at the specified clock network pins.

Normally when you perform latency reporting at intermediate clock network pins, the report is generated for all sequential cell clock pins in the transitive fanout of those pins. With the *-probe* option, the latency report is generated directly at the specified intermediate pins instead.

The *-probe* option is supported only for *-type latency* reporting; it is not supported for other reporting types.

Description

This command generates a report of clock timing information for the current design.

There are several reporting types provided, which allow you to examine skew, latency and transition time attributes of a given clock network or subnetwork at various levels of generality. By default, the report displays the values of these attributes only at sink pins (that is, the clock pins of sequential devices) of the clock network. You use the *-verbose* option to display source-to-sink path traces. If you specify several clock domains, the *report_clock_timing* command generates a separate subreport for each clock domain.

At the highest level of abstraction is the summary style report, which provides only a list of maxima and minima of the skew, latency, and transition time attributes over the given networks. At a lower level of abstraction are the transition, latency, and skew type reports, called list style reports, in which you can sort, filter and display the worst set of sink pins in the given network with respect to a single attribute of interest. For skew reports, each report entry is a pair of sink pins and their relative skew. For transition time or latency reports, each entry corresponds to a single sink pin. The lowest level of abstraction is provided by verbose mode, which replaces every sink pin in a list style report by a corresponding source-to-sink path trace.

In both summary and list style reports, the rightmost column is an attributed column. Corresponding to each sink pin, the set of character symbols in this column indicates the following:

```
Symbol    Meaning
------    -------
 r        Rising transition
 f        Falling transition
 p        Propagated clock to this pin
 i        Clock inversion to this pin
 -        Launching transition
 +        Capturing transition
 m        Library clock insertion delay
```

In verbose mode, back-annotations on path elements in the timing path are indicated using a character symbol. For definitions of these character symbols, see the manual page of the *report_timing* command.

Skews reported by *report_clock_timing* are local skews only. Local skew exists from one sink pin to another only if their associated sequential devices are connected via a data path in the appropriate from-to sense. Note that even if all data paths between the sequential devices are false because of user-defined exceptions, the skew still qualifies as local skew if the devices are connected topologically.

If detailed variation-aware clock analysis is enabled using the *variation_analysis_mode* variable, the *report_clock_timing* command evaluates the attributes of latency, transition time and skew as statistical entities. By default, the quantile of these attributes is used for both ranking and display purposes. The ranking can be controlled using the *-sort_by* option, while the display parameter can be changed from quantile to mean using the *variation_derived_scalar_attribute_mode* variable. To augment the display with additional statistical parameters of *report_clock_timing* attributes, see the *-variation* option.

Examples

The following example shows a typical summary style report.

```
pt_shell> report_clock_timing -type summary
****************************************
Report : clock timing
```

```
        -type summary
        -nworst 1
 ...
****************************************

  Clock: CK1
--------------------------------------------------------------------------------
---
  Maximum setup launch latency:
      flop9/CP                                                  3.08
 rp-+

  Minimum setup capture latency:
      or2_3/B                                                   1.15
 fpi-+

  Minimum hold launch latency:
      or2_3/B                                                   1.15
 fpi-+

  Maximum hold capture latency:
      flop9/CP                                                  3.08
 rp-+

  Maximum active transition:
      or2_3/B                                                   0.20
 fpi-+

  Minimum active transition:
      or2_3/B                                                   0.09
 fpi-+

  Maximum setup skew:
      flop9/CP
 rp-+
      flop10/CP                                                 0.87
 rp-+

  Maximum hold skew:
      flop9/CP
 rp-+
      flop10/CP                                                 -0.21
 rp-+
--------------------------------------------------------------------------------
---
```

The following example displays the five worst setup skews in the clock network of CLK1, taking uncertainty into account.

```
pt_shell> report_clock_timing -clock CLK1 -type skew -setup \\
            -nworst 5 -include_uncertainty_in_skew
****************************************
Report : clock timing
```

```
        -type skew
        -nworst 5
        -setup
        -include_uncertainty_in_skew
Design : multi_domain
 ...
****************************************

  Clock: CLK1

  Clock Pin                                Latency   Uncert    Skew
-------------------------------------------------------------------------
---
  f2_2/CP                                  6.11
 rp-+
  f2_1/CP                                  2.01      0.11      4.21
 fp-+

  l3_2/G                                   4.10
 rpi-
  f1_2/CP                                  1.00      0.22      3.32
 rpi-+

  l2_2/G                                   4.11
 rp-
  f1_2/CP                                  1.00      0.12      3.23
 rpi-+

  f2_2/CP                                  6.11
 rp-+
  l3_3/G                                   3.01      0.11      3.21
 rp-

  l1_3/G                                   5.11
 rp-
  f2_1/CP                                  2.01      0.11      3.21
 fp-+
-------------------------------------------------------------------------
---
```

The following displays the five worst launching latencies for hold paths in the clock network of CLK2.

```
pt_shell> report_clock_timing -clock CLK2 -hold -launch -nworst 5 \\
            -type latency
****************************************
Report : clock timing
        -type latency
        -launch
        -nworst 5
        -hold
Design : multi_domain
 ...
```

```
****************************************

  Clock: CLK2

                                                  --- Latency ---
  Clock Pin                       Trans     Source    Network   Total
-------------------------------------------------------------------------------
---
  f1_2/CP                         0.04      0.00      1.00      1.00
 rpi-+
  f1_3/CP                         0.00      0.01      1.00      1.01
 fp-+
  f2_1/CP                         0.01      0.01      2.00      2.01
 rp-+
  f1_1/CP                         0.00      0.00      3.00      3.00
 rpi-+
  f3_2/CP                         0.00      0.00      3.00      3.00
 fpi-+
-------------------------------------------------------------------------------
---
```

The following example demonstrates how to request a verbose report showing the worst local skew from f2_2/CP to any other sink pin.

```
pt_shell> report_clock_timing -type skew -verbose -from f2_2/CP
****************************************
Report : clock timing
         -type skew
         -verbose
         -nworst 1
         -setup
Design : multi_domain
 ...
****************************************

  Clock: CLK1

  Startpoint: f2_2 (rising edge-triggered flip-flop clocked by CLK1)
  Endpoint: f2_1 (rising edge-triggered flip-flop clocked by CLK1)

  Point                                  Trans       Incr       Path
  -------------------------------------------------------------------
  clock source latency                               0.11       0.11
  clk2 (in)                               0.00       0.00       0.11 r
  az_1/Z (B1I)                            0.09       1.00 H     1.11 r
  az_2/Z (B1I)                            0.13       1.00 H     2.11 r
  bf2_2_1/Z (B1I)                         0.02       1.00 H     3.11 r
  if2_2_1/Z (IVA)                         0.44       1.00 H     4.11 f
  bf2_2_2/Z (B1I)                         0.01       1.00 H     5.11 f
  if2_2_2/Z (IVA)                         0.13       1.00 H     6.11 r
  f2_2/CP (FD1)                           0.13       0.00       6.11 r
  startpoint clock latency                                      6.11
```

```
  clock source latency                                0.01       0.01
  clk2 (in)                              0.00         0.00       0.01 r
  az_1/Z (B1I)                           0.02         1.00 H     1.01 r
  az_3/Z (B1I)                           0.01         1.00 H     2.01 r
  f2_1/CP (FD1)                          0.01         0.00       2.01 r
  endpoint clock latency                                         2.01
  ------------------------------------------------------------------------
  startpoint clock latency                                       6.11
  endpoint clock latency                                        -2.01
  ------------------------------------------------------------------------
  skew                                                           4.10
```

The following example traces the two worst launch latencies for hold paths in the clock network of CLK1.

```
pt_shell> report_clock_timing -type latency -hold -verbose \\
            -nworst 2 -clock CLK1
****************************************
Report : clock timing
        -type latency
        -verbose
        -launch
        -nworst 2
        -hold
Design : multi_domain
 ...
****************************************

  Clock: CLK1

  Endpoint: f1_2 (rising edge-triggered flip-flop clocked by CLK1')

  Point                                  Trans       Incr       Path
  ------------------------------------------------------------------------
  clock source latency                                0.00       0.00
  clk1 (in)                              0.00         0.00       0.00 f
  if1_2_1/Z (IVA)                        0.04         1.00 H     1.00 r
  f1_2/CP (FD1)                          0.04         0.00       1.00 r
  total clock latency                                            1.00

  Endpoint: f1_3 (rising edge-triggered flip-flop clocked by CLK1)

  Point                                  Trans       Incr       Path
  ------------------------------------------------------------------------
  clock source latency                                0.01       0.01
  clk1 (in)                              0.00         0.00       0.01 r
  bf1_3_1/Z (B1I)                        0.00         1.00 H     1.01 r
  f1_3/CP (FD1)                          0.00         0.00       1.01 r
  total clock latency                                            1.01
```

The following example makes use of a slack filter to display the worst three skews between latches whose latch-to-latch paths are violating.

```
pt_shell> report_clock_timing -type skew -nworst 3 \\
            -slack_lesser_than 0.0
****************************************
Report : clock timing
        -type skew
        -slack_lesser_than 0.00
        -nworst 3
        -setup
Design : multi_domain
 ...
****************************************

  Clock: CLKA

  Clock Pin                     Latency    CRP      Skew      Slack
--------------------------------------------------------------------------
---
  f2_2/CP                       6.01
 rp-+
  f2_1/CP                       2.11      -0.03     3.87      -1.49
 rp-+

  l2_2/G                        4.01
 rp-
  f1_2/CP                       1.10      -0.21     2.70      -6.64
 rpi-+

  l1_3/G                        5.01
 rp-
  f2_1/CP                       2.11      -0.32     2.58      -1.63
 rp-+
--------------------------------------------------------------------------
---
```

The following example requests the two largest setup skews for paths both launched and captured by any clock networks in the list *$my_clocks*.

```
pt_shell> report_clock_timing -type interclock_skew \\
             -from_clock $my_clocks -to_clock $my_clocks \\
             -nworst 2 -include_uncertainty_in_skew -show_clocks
****************************************
Report : clock timing
        -type interclock_skew
        -nworst 2
        -setup
        -include_uncertainty_in_skew
        -show_clocks
 ...
****************************************

  Number of startpoint pins   : 907
  Number of endpoint pins     : 907
```

```
  Number of startpoint clocks : 5
  Number of endpoint clocks   : 5

  Clock Pin                                Latency   Uncert    Skew
-------------------------------------------------------------------------
---
  orig_clk
  orig_if/ram_pdp1/FFB35/CK                1016.72
 rp-+
  dcd_ram/ram_clk
  dcd_ram/dcd_ram/RAM/CKRB                 149.60    195.00    1062.12
 rp-+

  comp_clk
  comp_if/c_port_if/edge_trig_reg/CK       1400.42
 rp-+
  comp_clk
  comp_if/c_port_if/posi/iq_reg/CK         1159.09   199.00    440.34
 rp-+
-------------------------------------------------------------------------
---
```

The following reports the largest two setup skews in the domain of CLKB in the context of a detailed variation-aware clock analysis, and requests the augmentation of the report with variation information.

```
pt_shell> report_clock_timing -type skew -clock CLKB -variation
****************************************
Report : clock timing
         -type skew
         -nworst 2
         -variation
Design : multi_domain
 ...
****************************************

  Clock: CLKB

                                                 --- Skew ---
  Clock Pin                       Latency   Stddev    Mean      Quant
-------------------------------------------------------------------------
---
  f2_2/CP                         0.40221
 rp-+
  f2_1/CP                         0.13533   0.02825   0.26688   0.34732
 rp-+

  l2_1/CP                         0.33036
 rpi-+
  l2_2/CP                         0.18638   0.02626   0.14398   0.20948
 rp-+
```

Feedback

See Also

- report_clock
- report_timing
- set_clock_uncertainty
- report_default_significant_digits
- timing_remove_clock_reconvergence_pessimism

---

report_collection

Output a tabular report of attribute values for elements in a collection.

Syntax

report_collection

```
collection
[-type report_type]
[-header header_type]
[-max_rows value_count]
[-nopslit]
[-columns column_specification]

string collection
list report_type (
string header_type
int value_count
list column_specificaton
```

Arguments

collection

Specifies the collection on which to report. The collection may be homogeneous or heterogeneous. The report comprises a header with information about the report such as the application name and version and the date and a tabular report on attribute values for elements of the collection. The tabular report content is determined by the of report requested with the *-type* option. The attributes on which the report is generated is determined by the argument to the *-columns* option.

Feedback

`-type` *report_type*

This option accepts an argument of a list of valid report type names. A report type is one of the following values: "values" | "statistics" | "distribution". If the option is omitted, the default report type is "values". The report types may be combined in a list to output multiple reports in one run. Each invocation of the command produces a single report header output unless it is suppressed with the *-header* option.

The "values" report is a tabular output of attribute values for the collection elements, with one row of values per collection element. The report columns comprises formatted attribute values for the objects.

The "statistics" report type outputs the following statistical data about the attribute values

```
* Minimum, lower quartile, median upper quartile, and maximum
 values for numeric attributes.
* The mean and standard deviation for numeric attributes.
* Number of null attribute values for each attribute.
* Number of valid attribute values for each attribute.
* Number of unique attribute values for each attribute.
```

The "distribution" report type outputs a header section with information about the report, followed by the following statistical values about the data.

```
* Number of null attribute values for each attribute.
* Number of valid attribute values for each attribute.
* Number of unique attribute values for each attribute.
* Distribution of values with number of occurrences of each
 attribute value..
```

If both "statistics" and "distribution" reports are requested, the overlapping portion of the two reports is output only once.

All report types begin with a line displaying the collection composition including the size of the input collection followed by object count by object class type. When "statistcs" or "distribution" type report is included, the object count breakdown by object class type is always provided regardless of the size of the input collection. In a "values" only report, however, breakdown of object count by object class type is provided for a heterogeneous collection only when the collection contains one hundred or fewer items.

`-header` *header_type*

This option accepts one of the following valid argument values: "standard" | "none". If the option is omitted, the default header type is "standard". The "standard" header consists of the report name, command arguments, product name, product version, and the report date.

`-max_rows` *`value_count`*

This option indicates how many rows of data to include in the values and the value distribution tables. If the *collection* is very large, you may want to use the *-max_rows* option to limit the size of your report output. Number of text lines in the report may be greater than the *-max_rows* argument value if some data rows require multiple lines to output. This option value has no impact on the distribution and statistics calculations, for which the entire collection is processed. However, the *-max_rows* value, if given will limit the number of rows output for the value distribution table. If *-max_rows* value is given and truncates either the values or the distribution table output, a line is added to show the number of remaining items that were not shown.

`-nosplit`

This option indicates that the default column formatting used for a value is that is too long for the column width is "none", rather than the default "split". See the -columns option description below for more details.

`-columns` *`column_specification`*

Indicates the set of attributes on which to report and their order in the report. At a minimum, *column_specification* is a list of attribute names of elements of the collection.

The special attribute name "object_class" may be used to include the element object class names, such as "cell" or "net", in the report output.

If the option is omitted, then the object full names and object class (or type) names are output by default.

Each attribute name may be followed by a list of token-value pairs to indicate report column formatting directives. The value tokens for formatting are *width*, *precision*, *justify*, *label*, and *fit*. All column formatting specifications are optional.

width token is followed by a positive integer to value indicate the desired width of the column in number of characters. If not given, then the attribute value is queried for the first 100 elements of the collection and the longest value string determines the width of the column.

precision token is followed by a positive integer to value indicate the desired precision used to display floating point values (digits after the decimal point). If not given then the default precision for the attribute is used. This is ignored for non floating point attributes.

justify is followed by one of "left" or "right" to indicate the value's alignment in the column. If not given, then numeric attributes are justified right and other attributes are justified left.

Feedback

By default, the column headers show the name of the attribute. *label* is followed a string to display in the report column header in place of the attribute name.

fit is followed by one of the following valid argument values: "split" | "none" | "truncate" | "wrap" | "elide_left" | "elide_center" | "elide_right". This option indicates how to format an attribute value that is too long to fit in the column's width. If option is omitted, the default formatting is "split". However, if the option *-nosplit* is given, then the default formatting becomes "none". Explicit *fit* values given in a column specification overrides the default set by *-noSplit*.

The "split" formatting inserts a newline after the value which is too long and continues the tabular row on the next line, justifying the next value to the correct column.

The "none" formatting does nothing to reformat a value that is too long, continuing with the next column value. Therefore, the remainder of the tabular row will not be aligned with column headers.

The "truncate" formatting truncates the value to the width of the column, discarding the remainder of the value.

The "wrap" formatting truncates the value to the column and continues the rest of the tabular row on the same row. The remainder of the value is output on the following line(s), taking as many lines as is needed to finish outputting the rest of the value.

The "elide_left" formatting truncates the beginning of the value and adds the elision string "..." at the beginning of the value.

The "elide_center" formatting omits the middle of the value and inserts the elision string "..." at the center of the value.

The "elide_right" formatting truncates the end of the value and adds the elision string "..." at the end of the value.

Description

This command outputs a tabular report of attribute values for elements of the given collection. The attribute values in the report are determined by the list of attributes given as the *-columns* option argument. The columns in the tabular report will be ordered by order of attributes in the list. The command *list_attributes* may be called to see a list of attributes defined for an object class.

The *-column* option also accepts additional optional column formatting specifications in the attribute list. You may specify the header label and the width for each column, how to justify the value, and how to fit a value in a column when the value is too long to fit within the column width.

If the collection is large, the report size may be limited by indicating a *-max_rows* value. The commands *filter_collection* and *sort_collection* may be called to filter and sort a collection based on some criteria prior to creating a report for the collection elements.

Statistical and value distribution reports on the collection elements may be produced using the *-type* option arguments "statistics" and "distribution", respectively. Report types may be combined in a list.

The report header is followed by a line of report with collection size and object count by object data types. Precise object counts by types are always shown for "statistics" and "distribution" type reports. For "values" report, it is shown if the collection is homogenous or small. If the collection is heterogeneous and large, the object type shown for "values" report is "heterogeneous".

Examples

The following example from IC Compiler II creates a statistical report on a collection of cells along with a value distribution output.

```
icc2_shell> set cells [get_cells -hierarchical *]
icc2_shell> report_collection $cells -type {statistics distribution}
```

The following example, also from IC Compiler II, creates a report on a collection of cells with the full_name, area, and number_of_pins attribute values.

```
icc2_shell> set cells [get_cells U*]
icc2_shell> report_collection $cells -columns {full_name area
 number_of_pins}
```

The next example creates the same report but limits the width of the full_name column to 26 characters, designating "elide_center" formatting for full_name values that are longer than 26 characters.

```
icc2_shell> report_collection $cells -columns \
                              {{full_name {width 26} {fit elide_center}}
 \
                                area number_of_pins}
```

The next examples limits the output to the first 10 objects in the collection.

```
icc2_shell> report_collection $cells -columns \
                              {{full_name {width 26} {fit elide_center}}
 \
                                area number_of_pins} -max_rows 10
```

The next example first sorts the original collection of cells by the area attribute value in descending order, limiting the resulting collection to the top 10 area values. The example then careates a report for the resulting collection.

```
icc2_shell> set sorted_cells [sort_collection -dictionary \
                             $cells {area- full_name} -limit 10]
```

Feedback

```
icc2_shell> report_collection $sorted_cells -columns \
                              {{full_name {width 26} {fit elide_center}}
 \
                                area number_of_pins}
```

The next example reports on a sorted collection as in the above example but further limits the report to the first 10 objects in the collection. Note that this command may produce a report that is different from the above example. Limiting the sort to the first 10 values may have produced a collection with more than 10 objects since multiple objects may share the same area value.

```
icc2_shell> set sorted_cells [sort_collection -dictionary \
                             $cells {area- full_name} -limit 10]
icc2_shell> report_collection $sorted_cells -columns \
                              {{full_name {width 26} {fit elide_center}}
 \
                                area number_of_pins} -max_rows 10
```

See Also

- collections
- filter_collection
- sort_collection
- write_collection
- list_attributes

---

report_constraint

Displays constraint-related information about a design.

Syntax

status *report_constraint*

```
[-all_violators]
[-verbose]
[-path_type slack_only | end]
[-ignore_register_feedback feedback_slack_cutoff]
[-max_delay_groups]
[-min_delay_groups]
[-max_capacitance]
[-min_capacitance]
[-max_transition]
[-pulse_clock_max_transition]
[-min_transition]
[-pulse_clock_min_transition]
[-max_fanout]
```

```
[-min_fanout]
[-min_pulse_width]
[-pulse_clock_min_width]
[-pulse_clock_max_width]
[-min_period]
[-recovery_group]
[-removal_group]
[-max_skew]
[-clock_separation]
[-clock_gating_setup_group]
[-clock_gating_hold_group]
[-significant_digits digits]
[-nosplit]
[-connection_class]
[-boundary_check]
[-boundary_check_include boundary_checks]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-format style]
[-output file_name]
[-default_scenario_name scen_name]
[-waveform_integrity static | dynamic]
[-pba_mode none | path | exhaustive | ml_exhaustive]
[-include_hierarchical_pins]
[-sms_scenarios sms_scenarios_list]
[object_list]
```

Data Types

feedback_slack_cutoff	float
digits	integer
boundary_checks	list
pre_command_string	string
post_command_string	string
style	string
file_name	string
scen_name	string
object_list	list
sms_scenarios_list	collection

Arguments

`-all_violators`

Shows a summary of the worst violation per endpoint of each violated design rule constraint in the current design.

`-verbose`

Shows detailed information about each constraint violation. Multiple violations for a given constraint are listed from the largest to the smallest violation. If no objects are specified, the worst single violation is given for each constraint type.

`-path_type slack_only | end`

Specifies one of the following formats for the path report:

- *slack_only* (the default) - Shows only endpoint slacks.
- *end* - Shows the endpoint path total, required time, and slack for each path.

You cannot use the *-path_type* option together with the *-verbose* option.

`-ignore_register_feedback` *`feedback_slack_cutoff`*

Ignores any timing path that starts and ends at the same register and holds a value. This option applies to minimum delay as well as maximum delay reports. Only paths with slack less than the specified *feedback_slack_cutoff* are ignored. This option is applied as a filter to the paths after they are generated. Therefore, the number of paths generated might be less than the number specified with the *-nworst* and *-max_paths* options.

`-max_delay_groups`

Shows only the maximum delay and setup information. In the default report, the worst violation per clock group is displayed.

The maximum delay information shows cost by path group. This includes violations of setup time on registers or ports with output delay as well as violations of *set_max_delay* commands. The total maximum delay cost is the sum of the weighted cost of each group. For information about creating path groups, see the *group_path* man page. To see the current path groups in the design, use the *report_path_group* command.

`-min_delay_groups`

Shows only the minimum delay and hold information. In the default report, the sum of all violations per clock group is displayed.

The minimum delay cost includes violation of hold time on registers or ports with output delay as well as violations of the *set_min_delay* command.

`-max_capacitance`

Shows only the maximum capacitance constraint information. The *max_capacitance* design rule constraint limits the total capacitance on a net. The *-max_capacitance* option displays the *max_capacitance* cost (the sum of all maximum capacitance violations). To see details about the worst violation, use the *-verbose* and *-max_capacitance* options together. To see details about all *max_capacitance* violations, use the *-all_violators*, *-verbose*, and *-max_capacitance* options together. Note that the *max_capacitance* constraint is not checked for load pins.

`-min_capacitance`

Shows only the minimum capacitance constraint information. The *min_capacitance* design rule constraint limits the total capacitance on a net.

`-max_transition`

Shows only the maximum transition constraint information. The *max_transition* design rule constraint limits the transition time on pins and ports. If the library uses the cmos2 delay model, maximum edge rate information is shown instead. If the variable *timing_enable_slew_variation* is set to *true*, variation reporting is enabled for *max_transition*.

`-pulse_clock_max_transition`

Shows only the pulse clock maximum transition constraint information, similar to the *report_pulse_clock_max_transition* command.

`-min_transition`

Shows only the minimum transition constraint information. The *min_transition* design rule constraint specifies the minimum transition time on pins and ports. If the library uses the cmos2 delay model, maximum edge rate information is shown instead.

`-pulse_clock_min_transition`

Shows only the pulse clock minimum transition constraint information, similar to the *report_pulse_clock_min_transition* command.

`-max_fanout`

Shows only the maximum fanout constraint information. The *max_fanout* design rule constraint limits the fanout load on a net.

`-min_fanout`

Shows only the minimum fanout constraint information. The *min_fanout* design rule constraint specifies the minimum fanout load on a net.

`-min_pulse_width`

Shows only the minimum pulse width constraint information, similar to the *report_min_pulse_width* command. The *min_pulse_width* design rule constraint sets a minimum high or low pulse width at a clock pin or at pins or ports in the clock network.

`-pulse_clock_min_width`

Shows only the pulse clock minimum width constraint information. To enable pulse clock constraint checking, set the *timing_enable_pulse_clock_constraints* variable to *true*.

`-pulse_clock_max_width`

Shows only the pulse clock maximum width constraint information, similar to the *report_pulse_clock_max_width* command. To enable pulse clock constraint checking, set the *timing_enable_pulse_clock_constraints* variable to *true*.

`-min_period`

Shows only the minimum period constraint information. The *-min_period* option is a design rule used to set a minimum period on a clock signal.

`-recovery_group`

Shows only the recovery constraint information in default ***async_default*** group. The *recovery* type is a timing constraint used to describe the minimum allowable time between the control pin transition to the inactive state, and the active edge of the synchronous clock signal. This time is from the control signal going inactive to the clock edge that latches data in. The asynchronous control signal must remain constant during this time, or an incorrect value might appear at the outputs. In the default report, the worst violation in the default ***async_default*** group of the design is displayed.

`-removal_group`

Shows only the removal constraint information in default ***async_default*** group. The *removal* type is a timing constraint used to describe the minimum allowable time between the clock pin inactive edge, while the asynchronous pin is active, to the inactive edge of the same asynchronous control pin. In the default report, the sum of all violations in the default ***async_default*** group of the design is displayed.

`-max_skew`

Shows only the maximum skew constraint information. The *-max_skew* option is a timing constraint that checks the maximum separation time allowed between two clock signals.

`-clock_separation`

Shows only the clock separation constraint information.. The *-clock_separation* option is a timing constraint that checks the minimum separation time allowed between two clock signals.

`-clock_gating_setup_group`

Shows only the clock gating setup constraint information in default ***clock_gating_default*** group. The *clock_gating_setup* type is a timing constraint used to set a minimum setup time between a clock and a signal controlling the gating of that clock. In the default report, the worst violation in the default ***clock_gating_default*** group of the design is displayed.

`-clock_gating_hold_group`

Shows only the clock gating hold constraint information in default ***clock_gating_default*** group. The *clock_gating_hold* type is a timing constraint used to set a minimum hold time between a clock and a signal controlling the gating of that clock. In the default report, the sum of all violations in the default ***clock_gating_default*** group of the design is displayed.

`-significant_digits` *`digits`*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which has a default of 2. Use this option if you want to override the default.

`-nosplit`

Prevents line splitting to facilitate parsing of information from the report output.

`-connection_class`

Shows only the connection class constraint information. The connection_class constraint is displayed only if there is a connection_class violation. With the *-verbose* option, net pins and their associated connection class information are also displayed.

`-boundary_check`

Shows only HyperScale boundary constraint information. For more information, see the "Options for the HyperScale Flow" section.

`-boundary_check_include` *`boundary_checks`*

Reports the specified list of boundary checks. For more information, see the "Options for the HyperScale Flow" section.

`-pre_commands` *`pre_command_string`*

Specifies a string of commands to be executed in the worker context before the execution of merged reporting commands. Delimit commands with a semicolon (;). The maximum length of a command is 1000 characters. This option is available only if you invoke PrimeTime with the *-multi_scenario* option.

`-post_commands` *`post_command_string`*

Specifies a string of commands to be executed in the worker context after the execution of merged reporting commands. Delimit commands with a semicolon (;). The maximum length of a command is 1000 characters. This option is available only if you invoke PrimeTime with the *-multi_scenario* option.

`-format` *`style`*

Specifies the reporting format. The only allowed value is *csv*.

`-output file_name`

Specifies the output file name for CSV report.

`-default_scenario_name scen_name`

Specifies a scenario name in the CSV report. This option is valid only with *-format csv* and *-output* options. Note that in DMSA workers, this option overrides the implicit worker scenario name by the provided scenario name, in the CSV reports.

`-waveform_integrity static | dynamic`

Specifies the type of waveform integrity analysis:

- *static* - Available only if advanced waveform propagation is enabled.
- *dynamic* - Available only if signal integrity analysis is enabled.

To specify the constraints for waveform integrity analysis, use the *set_waveform_integrity_analysis* command.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Controls path-based analysis with the following modes:

- *none* (the default) - Path-based analysis is not applied.
- *path* - Path-based analysis is applied over the paths that gave the worst graph-based analysis violation. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- *exhaustive* - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.
- *ml_exhaustive* - Performs Machine Learning based exhaustive path-based analysis. This mechanism will deploy machine learning techniques to trade runtime vs accuracy during the design flow. Accuracy and runtime will match pba_mode exhaustive as the design approaches signoff.

`-include_hierarchical_pins`

Includes hierarchical pins in the timing report. This option is valid only when -verbose option is set to TRUE.

object_list

Specifies a list of pins, ports, or cells in the current design for which constraint related information is displayed. For the *-max_transition*, *-min_transition*, *-max_capacitance*, *-min_capacitance*, *-max_fanout*, or *-min_fanout* option, only pins and ports can be specified. For the *-boundary_check* option, only hierarchical cells can be specified.

The *-verbose* option can be used to report detailed constraint calculation, and *-all_violators* option can be used to report only violators among the supplied *object_list*. For the *-boundary_check* option, the *-all_violators* option can be used to report on all violating boundary checks in a HyperScale flow for a subset of instances.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only constraints compatible to the specified SMS scenarios collection are reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

By default, the *report_constraint* command displays the following information about all timing and design rule constraints checks on the current design:

- Whether the constraint is violated or met
- By how much the constraint value is violated (if it is violated).

The *object_list* can be optionally specified only with the *-max_transition*, *-min_transition*, *-max_capacitance*, *-min_capacitance*, *-max_fanout*, *-min_fanout*, and *-boundary_check* options. When an object list is specified, the *-verbose* option reports detailed constraint calculations for the objects, and *-all_violators* option reports only the violators among the objects.

Options for the HyperScale Flow

In a HyperScale flow (when the *hier_enable_analysis* variable is set to *true*), you can report boundary check violations that indicate whether a HyperScale block is used outside its context.

These checks are intended for use in a top-level analysis. They analyze the HyperScale blocks in the current analysis to ensure that the context used by block-level analysis bounds the actual conditions at the instance of that block in the top-level design. (In other words, they ensure that the top-level analysis is not using a block outside of the context it was analyzed at.)

To use these checks, the *timing_enable_hier_boundary_checks* variable must be set to *true* in both the block-level and top-level analysis runs.

The following options allow you to report boundary checks:

-boundary_check

Shows HyperScale boundary constraint differences between top-level and block-level analysis.

HyperScale boundary checks are not reported by the *report_constraint* command by default; this option must be used.

Boundary checks fall into two categories:

- Automatically fixable checks - HyperScale automatically adjusts the block-level boundary context during the next block-level run to resolve violations of these checks.
- Manually fixable checks – Violations of these checks indicate serious problems with the budgeted block-level constraints. HyperScale cannot automatically fix these violations; you must manually adjust the constraints.

In the report, an asterisk (*) next to a check name indicates an automatically fixable check. By default, only manually fixable checks are reported.

The *-all_violators* and *-verbose* options work with *-boundary_check* as follows:

- *-boundary_check* - the total number of check violations summed across all blocks is reported
- *-boundary_check -all_violators* - the total number of check violations for each violating block is reported
- *-boundary_check -verbose* - verbose details of the worst check violation of each type across all blocks is reported
- *-boundary_check -verbose -all_violators* - verbose details of all check violations are reported

-boundary_check_include *boundary_checks*

Changes the set of checks reported by the *-boundary_check* option.

Manually fixable checks can be specified with the following values:

```
boundary_ideal_network
clock_attributes
clock_mapping
clock_relations
clock_uncertainty
env_variables
global_timing_derate
library_mapping
operating_conditions
```

Automatically fixable checks can be specified with the following values:

```
boundary_logic_value
clock_latency
clock_skew_with_uncertainty
```

```
data_arrival
input_slews
```

The following special values include all manually fixable checks, all automatically fixable checks, or all checks:

```
non_auto_fixable
auto_fixable
all
```

If the *-boundary_check_include* option is not specified, only the manually fixable checks (*non_auto_fixable*) are reported.

For more information about a boundary check and how to resolve its violations, see the man page with "_violations" appended to the check name. For example, the following command shows information about *auto_fixable* violations:

```
pt_shell> man auto_fixable_violations
```

Examples

The following example shows brief constraint information for the current design:

```
pt_shell> report_constraint
****************************************
Report : constraint
Design : counter
 ...
****************************************

                                                  Weighted
    Group (max_delay/setup)       Cost     Weight     Cost
    ---------------------------------------------------
    CLK                           0.00      1.00      0.00
    default                       0.00      1.00      0.00
    ---------------------------------------------------
    max_delay/setup                                 0.00


    Constraint                                      Cost
    ---------------------------------------------------
    max_delay/setup                                 0.00  (MET)
    min_delay/hold                                  0.40  (VIOLATED)
    max_transition                                  0.00  (MET)
    max_fanout                                      0.00  (MET)
```

The following example displays detailed constraint information for the current design:

```
pt_shell> report_constraint -verbose
****************************************
Report : constraint
         -verbose
Design : counter
```

```
 ...
****************************************

  Startpoint: ffb (rising edge-triggered flip-flop clocked by CLK)
  Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)
  Path Group: CLK
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock CLK (rise edge)                    0.00       0.00
  startpoint clock skew (ideal)            0.00       0.00
  startpoint clock uncertainty             0.00       0.00
  ffb/CP (FD3)                             0.00       0.00 r
  ffb/QN (FD3)                             2.42       2.42 r
  w/Z (ND4)                                0.59       3.01 f
  q/Z (EO)                                 1.13       4.14 f
  j/Z (AO2)                                1.08       5.22 r
  ffd/D (FDS2)                             0.00       5.22 r
  data arrival time                                   5.22

  clock CLK (rise edge)                   10.00      10.00
  endpoint clock skew (ideal)              0.00      10.00
  endpoint clock uncertainty               0.00      10.00
  ffd/CP (FDS2)                            0.00      10.00 r
  library setup time                      -0.90       9.10
  data required time                                  9.10
  -----------------------------------------------------------
  data required time                                  9.10
  data arrival time                                  -5.22
  -----------------------------------------------------------
  slack (MET)                                        3.88


    Design: counter

    max_area              30.00
  - Current Area          78.00
  ------------------------------
    Slack                -48.00  (VIOLATED)
```

The following example displays detailed information on only the worst violation per endpoint for those constraints that have violations:

```
pt_shell> report_constraint -all_violators -verbose
****************************************
Report : constraint
         -all_violators
         -verbose
 ...
****************************************

  Startpoint: b (input port)
```

```
 Endpoint: z5 (output port)
 Path Group: default
 Path Type: max

 Point                                    Incr       Path
 -----------------------------------------------------------
 input external delay                     0.00       0.00 r
 b (in)                                   0.00       0.00 r
 U5/Z (IV)                                1.32       1.32 f
 U3/Z (NR2)                               3.35       4.67 r
 U18/Z (AO6)                              0.73       5.40 f
 U22/Z (AO4)                              1.42       6.82 r
 z5 (out)                                 0.00       6.82 r
 data arrival time                                   6.82

 max_delay                                6.50       6.50
 output external delay                    0.00       6.50
data required time                                  6.50
 -----------------------------------------------------------
 data required time                                  6.50
 data arrival time                                  -6.82
 -----------------------------------------------------------
 slack (VIOLATED)                                   -0.32


 Startpoint: c (input port)
 Endpoint: z3 (output port)
 Path Group: default
 Path Type: max

 Point                                    Incr       Path
 -----------------------------------------------------------
 input external delay                     0.00       0.00 r
 c (in)                                   0.00       0.00 r
 U6/Z (IV)                                1.34       1.34 f
 U2/Z (NR2)                               3.35       4.69 r
 U15/Z (AO7)                              0.87       5.56 f
 U24/Z (AO3)                              1.02       6.57 r
 z3 (out)                                 0.00       6.57 r
 data arrival time                                   6.57

 max_delay                                6.50       6.50
 output external delay                    0.00       6.50
 data required time                                  6.50
 -----------------------------------------------------------
 data required time                                  6.50
 data arrival time                                  -6.57
 -----------------------------------------------------------
 slack (VIOLATED)                                   -0.07


   Net: a
```

```
    max_fanout              5.00
  - Fanout                  7.00
  ------------------------------
    Slack                  -2.00   (VIOLATED)
```

The following example shows how to report all max transition violations:

```
pt_shell> report_constraint -max_transition -all_violators
****************************************
Report : constraint
         -all_violators
         -path_type slack_only
         -max_transition
 ...
****************************************

   max_transition

                       Required        Actual
   Pin                 Transition     Transition       Slack
   -------------------------------------------------------------
   CK2                    0.10          10.00          -9.90   (VIOLATED)
   m0/B                   0.10          10.00          -9.90   (VIOLATED)
   fc/D                   0.10           1.75          -1.65   (VIOLATED)
```

The following example shows how to report all minimum pulse width violations:

```
pt_shell> report_constraint -min_pulse_width -all_violators
****************************************
Report : constraint
         -all_violators
         -min_pulse_width
 ...
****************************************

   sequential_clock_pulse_width

                       Required       Actual
   Pin                 pulse width    pulse width    Slack
   ------------------------------------------------------------------------
   ff1/CP                 10.20          10.00          -0.20   (VIOLATED)
   ff2/CP                 10.20          10.04          -0.16   (VIOLATED)
   ff3/CP                  2.10           2.00          -0.10   (VIOLATED)
   ff3/CP                  2.10           2.00          -0.10   (VIOLATED)
   ff2/CP                 10.00           9.96          -0.04   (VIOLATED)


   clock_tree_pulse_width

                       Required       Actual
   Pin                 pulse width    pulse width    Slack
   ------------------------------------------------------------------------
   nand1/Z                10.00           9.58          -0.42   (VIOLATED)
```

```
   or1/B                10.00           9.58          -0.42   (VIOLATED)
   nand1/A              10.20          10.00          -0.20   (VIOLATED)
   or1/Z                10.20          10.04          -0.16   (VIOLATED)
   or1/Z                10.00           9.96          -0.04   (VIOLATED)
```

The following example displays how to report all max_skew violations:

```
pt_shell> report_constraint -max_skew -all_violators
****************************************
Report : constraint
        -all_violators
        -path_type slack_only
        -max_skew
 ...
****************************************

   max_skew
                        Required       Actual
   Pin                  pulse width    pulse width    Slack
   ------------------------------------------------------------------------
   ff3/c (r->f)         0.15           7.46          -7.31   (VIOLATED)
   ff1/c (r->f)         0.15           5.47          -5.32   (VIOLATED)
   ff2/c (f->f)         0.14           2.32          -2.18   (VIOLATED)
   ff2/c (r->r)         0.12           1.18          -1.06   (VIOLATED)
   ff4/c (f->f)         0.14           0.84          -0.70   (VIOLATED)
   ff4/c (r->r)         0.12           0.42          -0.30   (VIOLATED)
```

The following example displays how to report detailed connection_class violations:

```
pt_shell> report_constraint -connection_class -verbose
****************************************
Report : constraint
        -verbose
 ...
****************************************

    Net: en

    Pin                                 Connection_class
 ------------------------------------------------------------------------
                           User defined              Library defined
 ------------------------------------------------------------------------
   en                      xyz
   U18/B                                             default
   U20/B                                             default
   U22/B                                             default
   U14/B                                             default
   U16/A                                             default

    connection_class       0.00
  - Cost                   6.00
  ------------------------------
    Slack                 -6.00   (VIOLATED)
```

The following example reports all violated HyperScale block boundary checks for a specific hierarchical block:

```
pt_shell> report_constraint -boundary_check -boundary_check_include {all}
 \\
            -all_violators -verbose \\
            [get_cell BLK2]
 ...
HyperScale constraints report
Constraint: clock_uncertainty

Instance Type   From Clock   To Clock     Block  Top     Slack
-----------------------------------------------------------------
BLK2     setup  clk (rise)   clk (rise)   0.000  0.200  -0.200
BLK2     setup  clk (rise)   clk (fall)   0.000  0.200  -0.200
BLK2     setup  clk (fall)   clk (rise)   0.000  0.200  -0.200
BLK2     setup  clk (fall)   clk (fall)   0.000  0.200  -0.200
```

See Also

- create_clock
- group_path
- report_clock
- report_design
- report_min_pulse_width
- report_path_group
- report_pulse_clock_max_transition
- report_pulse_clock_max_width
- report_pulse_clock_min_transition
- report_pulse_clock_min_width
- report_timing
- set_max_delay
- set_waveform_integrity_analysis
- auto_fixable_violations
- non_auto_fixable_violations
- report_default_significant_digits
- timing_enable_pulse_clock_constraints

Feedback

- timing_enable_slew_variation
- timing_max_capacitance_derate
- timing_max_transition_derate

---

report_context

Reports the characterized timing context information.

Syntax

string *report_context*

```
[-timing]
[-environment]
[-design_rules]
[-constant_inputs]
[-nosplit]
[-multi_instance_merge_compatibility]
[-list_context_override]
[-margin]
[-include type_list]
[-noise]
object_list
```

Data Types

object_list	list
type_list	list

Arguments

`-timing`

Reports timing information, such as clocks, input and output delays, and timing exceptions.

`-environment`

Reports environment-related information, such as operating conditions (process, temperature, and voltage), wire load model, capacitive loads on input and output pins, and driving cell information on input pins.

`-design_rules`

Reports characterized design rule information, such as *max_capacitance*, *max_transition*, and *max_fanout*.

`-constant_inputs`

Reports logic constants propagated to input pins.

`-nosplit`

Does not split lines if a column overflows.

`-multi_instance_merge_compatibility`

Specifies clustering information within each multi-instance group that satisfies clock compatibility. This option can only be used for the HyperScale Constraint Extractor or characterize context flows. For multi-instance configurations, it checks if the clock constraints of different instances are compatible or not. If the clock constraints are not compatible, this means that those instances should not be put in the same group. It provides clustering information within each group that satisfies clock compatibility within each cluster. When this option is used, this information is printed as part of the report. A new column is added for this option which shows the cluster. A number shows the cluster number for each instance in that group. If the instance is not part of a multi-instance configuration, -1 is show as the cluster.

`-list_context_override`

This option enables reporting of context overriding exceptions in a HyperScale block-level analysis. Automatic override of user-specified I/O budgets in the block constraints with accurate HyperScale top-level context is critical for the HyperScale flow to achive flat-analysis accuracy in a block-level analysis. However, there are multiple reasons why this automatic context overriding process might either be skipped, or only succeed partially, or completely fail.

The primary causes of context override failure are:

- User directives applied by *set_dont_override*
- Error conditions, such as clock mapping errors
- Netlist change on port nets

This report lists all the ports whose context override was not fully successful, along with the reason(s).

If there is no top-level HyperScale context available for the current block-level run, no ports will be automatically overriden, and therefore no ports will be listed by this report. In this case, a message indicating there is no context loaded for the current design is issued.

The 'Not Overridden Reason' column lists the reasons why ports have no context override when there is context available. The reasons include:

- *dont_override_timing* - port with user-applied *set_dont_override* on timing
- *dont_override_noise* - port with user-applied *set_dont_override* on noise

- *net_mismatch* - port with netlist mismatch on port net
- *unresolved* - port context refers to clock(s) with mapping errors

`-margin`

This option enables reporting of context margin set by the *set_context_margin* command. The margin allocation mode annotation is:

- *s* - *pin_slack*: set by *set_context_margin -allocation pin_slack*.
- *c* - *clock_period* - set by *set_context_margin -allocation clock_period*.
- *o* - *usr_override* - set by *set_context_margin -allocation user_override*.

Where different margin settings overlap in scope, the more specific setting is reported.

`-include` *`type_list`*

This option allows you to report only particular types of context information. Specify a list of one or more of the following types:

- *input_delay*: report only the input delay of the context
- *output_delay*: report only the output delay of the context
- *clock_latency*: report only the clock_latency of the context

`-noise`

This option enables reporting of context noise.

`object_list`

Lists cells or pins/ports for which to report the context.

Description

Reports the timing context captured using the *characterize_context* command for the specified list of instances or pins.

This command can also be used to report context data loaded by the *read_context* command. Use *current_design* as the object if reporting all loaded context. Specify port objects to report the context loaded for those ports.

Command options allow you to specify which categories of context information to report. All information categories are reported if no option is specified.

Examples

The following command reports the timing-related context information for instance I1 and I2. The report shows clocks and their waveforms, point-to-point timing exceptions, input external delays, and output external delays.

Feedback

The input and output delay information lists the minimum rise, minimum fall, maximum rise, and maximum fall delays. Input and output delay information lists the clock to which the delay is relative. If "(f)" follows the clock name, the delay is relative to the falling edge of the clock; otherwise the delay is relative to the rising edge. If "(l)" follows the clock name, input delay is considered as coming from a level-sensitive latch, or output delay is considered as going to a level-sensitive latch. The default is that the delay is relative to a rising edge-triggered device.

Multiple output delays are listed for a characterized output port if it has more than one fanout or if the logic it drives terminates in more than one type of latches (that is, edge-triggered (falling or rising clock) or level-sensitive with falling or rising clock).

```
pt_shell> report_context -timing {I1 I2}
```

The following command reports the environment and design rule context information. The report shows design rule checking, wire load models, drive information of input pin, and capacitive load on the input and output pins of I2.

```
pt_shell> report_context -env -design_rules I2
```

The following command reports the context on port "in" loaded by the *read_context* command.

```
pt_shell> report_context in
```

The following command reports all context data loaded by the *read_context* command.

```
pt_shell> report_context [current_design]
```

See Also

- characterize_context
- write_context

---

report_coupling_capacitors

Reports the coupling capacitors for specified nets.

Syntax

status *report_coupling_capacitors*

```
[-verbose]
[-of_objects nets]
[-from node1 -to node2]
```

Feedback

Data Types

nets	list
node1	string
node2	string

Arguments

`-verbose`

Provides additional information about the coupling capacitors.

`-of_objects` *nets*

A list of one or more nets for which to report coupling capacitors. At least one net is required. Glob-style wildcards (*) are supported.

This option is mutually exclusive with the *-from* and *-to* options.

`-from` *node1*

Specifies a pin, port, or net internal node. The tool reports the coupling capacitors between this node and the node specified in the *-to* option. Both nodes must both belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

`-to` *node2*

Specifies a pin, port, or net internal node. The tool reports the coupling capacitors between this node and the node specified in the *-from* option. Both nodes must both belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

Description

This command reports the coupling capacitors for specified nets.

You must use either the *-of_objects* option or both the *-from* and *-to* options.

The default report contains a section for each victim net (the nets specified in the command arguments). The victim net heading lists the total coupling capacitance for the victim net. For each aggressor net, the report lists the total coupling capacitance between the aggressor net and the victim net and its percentage with respect to the total coupling capacitance on the victim net.

The *-verbose* report also contains a section for each victim net. It provides detailed information about the individual coupling capacitances between nodes of the victim net and nodes of the aggressor nets.

This command is supported only for transistor-level GPDs.

Feedback

Examples

The following example shows a default coupling capacitor report.

```
pt_shell> report_coupling_capacitors -of_objects SUM0
================================
Net: SUM0
Total capacitance: 0.013721
Report Type: Aggressors, summary
================================
Total CCAP      %Cc/Ct         Aggressor Net
==========      ======         =============
0.000925        6.741491       B0
0.000908        6.617593       A0
0.000468        3.410830       CIN
```

The following example shows a verbose coupling capacitor report. Net SUM0 has two pins, 0/33/M2/s and 0-/33/M-1/s, which can be determined with the *get_pins* command.

```
pt_shell> get_pins -of [get_nets SUM0]
{"0/33/M2/s", "0/33/M1/s"}
pt_shell> report_coupling_capacitors -of_objects SUM0 -verbose
================================
Net: SUM0
Total capacitance: 0.013721
Report Type: Aggressors, detailed
================================
Victim Node Victim Lyr Aggressor Node Aggressor Lyr Capacitance %Cc/Ct
=========== ========== ============== ============= =========== ========
0/33/M2/s   SUBSTRATE  A0:24          metal1        0.000299    2.179141
SUM0:5      metal2     A0:24          metal1        0.000047    0.342541
0/33/M2/s   SUBSTRATE  A0:25          metal1        0.000060    0.437296
SUM0:5      metal2     A0:25          metal1        0.000502    3.658625
0/33/M2/s   SUBSTRATE  B0:25          metal1        0.000370    2.696596
SUM0:4      metal2     B0:25          metal1        0.000001    0.007288
```

---

report_crpr

Reports the clock reconvergence pessimism (CRP) calculated between specified register clock pins or ports.

Syntax

string *report_crpr*

```
-from from_latch_clock_pin
-to to_latch_clock_pin
[-from_clock from_clock]
[-to_clock to_clock]
[-setup | -hold]
[-significant_digits digits]
[-sms_scenarios sms_scenarios_list]
```

Feedback

Data Types

from_latch_clock_pin	string
to_latch_clock_pin	string
from_clock	string
to_clock	string
digits	integer
sms_scenarios_list	collection

Arguments

`-from` *from_latch_clock_pin*

Specifies the clock pin of the launching sequential device or clock gating check to be reported. A constrained input port can also be used.

`-to` *to_latch_clock_pin*

Specifies the clock pin of the capturing sequential device or clock gating check to be reported. A constrained output port can also be used.

`-from_clock` *from_clock*

Specifies the clock that fans out to the launching sequential device.

`-to_clock` *to_clock*

Specifies the clock that fans out to the capturing sequential device.

`-setup`

Indicates that the CRP used for a setup data path is to be reported. The *-setup* and *-hold* options are mutually exclusive. If neither option is specified, *-setup* is assumed.

`-hold`

Indicates that the CRP used for a hold data path is to be reported. The *-setup* and *-hold* options are mutually exclusive. If neither option is specified, *-setup* is assumed.

`-significant_digits` *digits*

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13.

If this option is not specified, the number of significant digits is determined by the *report_default_significant_digits* variable.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only data compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Feedback

Description

Reports the clock reconvergence pessimism (CRP) calculated between specified register clock pins or ports.

This command displays the following information about the calculation of the CRP between the two specified sequential elements:

- The name of the common pin in the clock network.
- The clock edges (rising or falling) that propagate through the common point to the launching and capturing registers.
- The value of the *timing_crpr_threshold_ps* variable.
- A tabulation of the arrival times for both clock edges at the common point and their associated CRP (crp_rise and crp_fall).
- The CRP used along with the reasoning behind the selection of the particular CRP (crp_rise or crp_fall) used for that report.
- The CRP value used is independent of the CRPR threshold, and, therefore, is referred to as the exact CRP. The *report_timing* and *report_clock_timing* commands report a CRP value that might be pessimistic by up to the value of the CRPR threshold. If the CRP value used by the *report_timing* or *report_clock_timing* command differs from the exact CRP, this value is also reported.

If the *timing_remove_clock_reconvergence_pessimism* variable is set to *false*, CRPR is inactive and this command does not work.

To generate a CRP report for specific clocks, use the *-from_clock* and *-to_clock* options. Otherwise, the tool reports all clocks that fanout to each sequential device.

For each potential common point in the design, the tool calculates

- crp_rise from rising arrival times at the common point
- crp_fall from falling arrival times at the common point

The value of CRP used in the report is dependent on what clock edges propagate through the common point to the clock pins of the launching and capturing devices. The clock edges are included in the CRPR report as "Launching edge at common point", "Capturing edge at common point".

For example, if a rising edge through the common point triggers the launching device and also triggers the capturing device, then a rising CRP value (crp_rise) is used. Similarly, for a falling edge propagating through the common point and subsequently launching and capturing data, a falling CRP value is used (crp_fall).

If a rising edge through the common point launches the data, and a falling edge through the common point captures it, a mismatch in sense occurs. In this case, if the *timing_clock_reconvergence_pessimism* variable is set to

- *normal*, the minimum of crp_rise and crp_fall is used
- *same_transition*, the CRP is reported as zero

This selection process is reported in the selection details section of the report.

If either clock edge is reported as "RISING or FALLING", there is a non-unate path between the common point and that clock pin, such that botch edge directions result in active clock edges at the clock pin.

If dynamic information is enabled, two sets of arrival totals are tabulated. Dynamic information can result from SI delta delays, dynamic clock latencies or dynamic rail voltages. In this case, the two sets of arrival totals tabulated are those computed with and without dynamic information being available. The arrival totals computed without dynamic information are equivalent to those used in a PrimeTime analysis with no dynamic information available.

The additional set of arrivals result in four CRP values from which the CRP value used should be chosen. These four CRP values can be summarized as rise/fall CRP and rise/fall dynamic CRP. The rise/fall decision is made based on clock edges incident at the clock edge as before. The decision over whether or not to use dynamic CRP is based on the temporal separation of launching and capturing clock edges. It is only valid to use dynamic CRP if the clock edge launching and capturing data occurs at the same time.

In the case when the capturing sequential device is a level-sensitive latch, two CRP values are reported. The first corresponding to the opening edge of the latch (this CRP also appears in the timing report) and the other corresponding to the closing edge of the device. The selection details section also reports the decision making process behind both CRP values. Note that in this case, since the opening and closing clock edges at the latch are always different, there is always a mismatch in clock edges for one of them.

Examples

The following example displays how to use the command to report the clock reconvergence pessimism calculated between the sequential elements, ffa and ffd, for a setup data path.

```
pt_shell> report_crpr -from ffa/CP -to ffd/CP -setup

report_crpr -from ffa/CP -to ffd/CP -setup
****************************************
Report : CRP Calculation
Design : counter
 ...
****************************************
```

```
  Startpoint: ffa (rising edge-triggered flip-flop clocked by CLK)
  Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)

  Common Point: CLK <inside>
  Common Clock: CLK
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold: 0.02

  Arrival Times                        Early    Late     CRP
-------------------------------------------------------------
  Rise                                  3.00    19.00    16.00
  Fall                                  5.00    23.00    18.00
-------------------------------------------------------------

  Selection Details
-------------------------------------------------------------
  Edge Match:          Match, using rise CRP
-------------------------------------------------------------
  clock reconvergence pessimism                          16.00
```

The following example displays a report where the launching device is a flip-flop and the capturing device is a latch. Note that in this case the mismatch in clock edges occurs for the CRP corresponding to the closing edge at the latch.

```
pt_shell> report_crpr -from reg1/CP -to lat2/G
****************************************
Report : CRP Calculation
Design : borrow
 ...
****************************************

  Startpoint: reg1 (rising edge-triggered flip-flop clocked by clk)
  Endpoint: lat2 (positive level-sensitive latch clocked by clk)

  Common Point: clk_buf/Z
  Common Clock: clk
  Launching edge at common point: RISING
  Latch open edge at common point: RISING
  CRPR threshold: 0.01

  Arrival Times                        Early    Late     CRP
-------------------------------------------------------------
  Rise                                2.1229   2.6529   0.5300
  Fall                                2.0868   2.7868   0.7000
-------------------------------------------------------------

  Selection Details
-------------------------------------------------------------
  Edge Match (opening): Match, using rise CRP
  Edge Match (closing): Mismatch, using min(rise CRP, fall CRP)
-------------------------------------------------------------
```

```
  clock reconvergence pessimism (open edge)           0.5300
  clock reconvergence pessimism (close edge)          0.5300
```

The following example displays the case when both SI and CRPR are enabled. As outlined in a previous paragraph, two sets of arrival totals are computed in this case (for example, when both CRPR and SI are enabled). One set of arrival totals are computed with no SI information and another are computed considering SI information. To illustrate this, both a setup and hold report are displayed to highlight the case when CRP is calculated from arrival times without delta delays (setup) and the case when CRP is calculated from arrival times including delta delays (hold with the launching and capturing clock edge occurring at the same time).

```
pt_shell> report_crpr -from seg1/u3/CK -to seg1/u9/CK
****************************************
Report : CRP Calculation
 ...
****************************************

  Startpoint: seg1/u3 (rising edge-triggered flip-flop clocked by CLK1)
  Endpoint: seg1/u9 (rising edge-triggered flip-flop clocked by CLK1)

  Common Point: seg1/u17_c/Y
  Common Clock: CLK1
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold: 0.02

  Arrival Times (Static)              Early    Late     CRP
-------------------------------------------------------------
  Rise                               0.3633   0.3746   0.0113
  Fall                               0.3871   0.3991   0.0120
-------------------------------------------------------------

  Arrival Times (Dynamic)             Early    Late     CRP
-------------------------------------------------------------
  Rise                               0.2142   0.3746   0.1604
  Fall                               0.2296   0.3991   0.1695
-------------------------------------------------------------

  Selection Details
-------------------------------------------------------------
  Arrival Times:       Static
  Edge Match:          Match, using rise CRP
-------------------------------------------------------------
  clock reconvergence pessimism                       0.0113


  Range of accuracy of CRP in report_timing, due to value
  of timing_crpr_threshold_ps:    0.0000 <= CRP <= 0.0113
```

Feedback

```
pt_shell> report_crpr -from seg1/u3/CK -to seg1/u9/CK -hold
****************************************
Report : CRP Calculation
 ...
****************************************

  Startpoint: seg1/u3 (rising edge-triggered flip-flop clocked by CLK1)
  Endpoint: seg1/u9 (rising edge-triggered flip-flop clocked by CLK1)

  Common Point: seg1/u17_c/Y
  Common Clock: CLK1
  Launching edge at common point: RISING
  Capturing edge at common point: RISING
  CRPR threshold: 0.02

  Arrival Times (Static)              Early    Late     CRP
-------------------------------------------------------------
  Rise                               0.3633   0.3746   0.0113
  Fall                               0.3871   0.3991   0.0120
-------------------------------------------------------------

  Arrival Times (Dynamic)             Early    Late     CRP
-------------------------------------------------------------
  Rise                               0.2142   0.3746   0.1604
  Fall                               0.2296   0.3991   0.1695
-------------------------------------------------------------

  Selection Details
-------------------------------------------------------------
  Arrival Times:       Static
  Edge Match:          Match, using rise CRP
-------------------------------------------------------------
  clock reconvergence pessimism                        0.1604
```

See Also

- set_clock_latency
- set_rail_voltage
- si_enable_analysis
- timing_clock_reconvergence_pessimism
- timing_crpr_threshold_ps
- timing_remove_clock_reconvergence_pessimism

---

report_ctpm

Report the CTPM in current design, for a library, or a certian CTPM file.

Feedback

Syntax

string *report_ctpm*

```
[-ctpm_file ctpm_file]
[-library lib]
[-verbose]
```

Data Types

```
ctpm_file       string
lib             collection
```

Arguments

`-ctpm_file ctpm_file`

Specifies the CTPM file to report. Without the option, CTPM used in current design will be reported.

`-library lib`

Specifies the library to report ML-CTPM.

`-verbose`

Verbose report for flattened per-lib_timing_arc CTPM values for ML-CTPM.

Description

The *report_ctpm* command reports CTPM related information used in PrimeShield SPICE2Design feature. This feature enables analysis for timing and power impact assessment of a given target. This target can be defined through a spice model representing next PDK, spot model, different spice corner.

Through ML, Project Sicily aims to bridge the gap between the base and the target by capturing and gap through spice process parameters placed inside of a Compact Timing Power Model (CTPM) database. When CTPM is consumed in a base STA session, the QoR of this session is shifted such that to match QoR of the target.

Examples

The following script reports CTPM information of a CTPM file.

```
set ps_enable_analysis true
set ps_enable_spice2design_analysis true

pt_shell> report_ctpm -ctpm_file lib1.ctpm -verbose
Information: Load file cwd/lib1.ctpm successfully.
Information: CTPM for base library lib1 START
base_process_label: N/A
base_custom_label: N/A
target_process_label: N/A
target_custom_label: N/A
```

```
rise per-arc CTPM value:
lib_cell,input_lib_pin,output_lib_pin,sense,when_condition pmos_lvt_vt
 nmos_lvt_vt pmos_lvt_ids0 nmos_lvt_ids0
AO211_NOM_D0_L,B,Z,positive_unate,"!A1&A2&!C" 0.0115144 0.011261 0.226212
 0.136295
INV_SKF_D18_L,I,ZN,negative_unate,default 0.000309105 0.011094
 -0.00795562 0.00556735
DFQN_NOM_D0_L,CP,QN,rising_edge,default 0.00909024 0.00304372
 -0.000555002 -0.000838983
fall per-arc CTPM value:
lib_cell,input_lib_pin,output_lib_pin,sense,when_condition pmos_lvt_vt
 nmos_lvt_vt pmos_lvt_ids0 nmos_lvt_ids0
AO211_NOM_D0_L,B,Z,positive_unate,"!A1&A2&!C" 0.00791495 0.0113072
 -0.602825 0.0921121
INV_SKF_D18_L,I,ZN,negative_unate,default 0.0114111 0.0010617 -0.00331055
 -0.00568498
DFQN_NOM_D0_L,CP,QN,rising_edge,default 0.0107667 0.00326676 -0.00439699
 -0.0056057
Information: CTPM for base library lib1 END

1
```

See Also

- gen_ctpm
- read_ctpm
- ps_enable_spice2design_analysis

---

report_delay_calculation

Displays the actual calculation of a cell or net timing arc delay value.

Syntax

int *report_delay_calculation*

```
[-min | -max]
[-from from_pin]
[-to to_pin]
[-of_objects objects
[-nosplit]
[-from_rise_transition value]
[-from_fall_transition value]
[-thresholds]
[-crosstalk]
[-derate]
[-sms_scenarios sms_scenarios_list]
[-sensitivity]
```

Feedback

Data Types

from_pin	string
to_pin	string
objects	collection
value	float
sms_scenarios_list	collection

Arguments

`-min`

Shows the minimum delay calculation. The design must be in either min or max mode.

`-max`

Shows the maximum delay calculation. This is the default if neither the *-min* nor *-max* option is specified.

`-from` *from_pin*

Specifies the startpoint of a timing arc within a design. For a cell timing arc, the pins must represent the input and output pins of a common leaf cell that have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. Port names are allowed in place of pin names for net arcs. Use either the *-from* and *-to* options together, or use the *-of_objects* option. However, you cannot set both.

`-to` *to_pin*

Specifies the endpoint of a timing arc within a design. For a cell timing arc, the pins must represent the input and output pins of a common leaf cell that have a timing arc specified between them in the library. For a net timing arc, the pins must be a driver and a load on a common net. Port names are allowed in place of pin names for net arcs. Use either the *-from* and *-to* options together, or use the *-of_objects* option. However, you cannot set both.

`-of_objects` *objects*

Specifies a collection of timing arcs (created with the *get_timing_arcs* command) on which to report. Arcs in the list are reported in order of from and to pins. Use either the *-from* and *-to* options together, or use the *-of_objects* option. However, you cannot set both.

`-nosplit`

Prevents line-splitting to facilitate parsing of information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`-from_rise_transition` *value*

Specifies a value to be used by the delay calculation for the from rise transition.

`-from_fall_transition` *value*

Specifies a value to be used by the delay calculation for the from fall transition.

`-thresholds`

Reports the characterization thresholds that are used for delay calculation.

`-crosstalk`

Reports the crosstalk information for a net arc. The arc is specified with the *-from_pin* and *-to_pin* options. It is not permitted with the *-of_objects* option and the user-chosen transition time *-from_rise_transition* and *-from_fall_transition* options. The crosstalk information is reported from the last *update_timing* command.

`-derate`

Reports derating components and derated delay.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only delay information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

`-sensitivity`

Reports sensitivity caused by physical parameter shift from PrimeShield features like SPICE2Design CTPM, PVT explorer condition what-if PPA analysis, device parameter margin, etc).

Description

This command provides detailed timing calculation information about the specified cell or net timing arc. This information is useful for debugging or verifying timing data in a technology library.

Both operating conditions and wire load models are considered when making delay calculations. Timing ranges as affected by the *set_timing_derate* command (similar to the *set_timing_range* dc_shell command) are not considered because they typically apply to an entire path.

The information reported is solely for the input arcs. Even if there are other arcs merging at the to-pin of an input arc, the transition time reported for the to-pin is the transition time that the input arc contributes to the merging point. As a result, the reported transition time might not match the transition time at the to-pin because the final transition time is the result of merging transition times contributed by all the input arcs. To get the final transition

time at the to-pin, either all these input arcs should be reported or an attribute access to the to-pin should be used.

The *-crosstalk* option on net arc, reports the aggressor and victim information for both rise and fall. It prints coupling capacitance, the driving library cell, the clocks reaching the net (both victim and aggressor). The aggressors have "Switching Bump", which is used for prioritizing the aggressors, and also have aggressor attributes. These informations are already available as net or design attributes. For example, "Switching Bump" is available as the net attribute si_xtalk_bumps. Considering the memory usage, only the worst case bumps for the net is stored, even though during the crosstalk delay calculation switching bump per net arc is used. "Switching Bump" is crosstalk delay bump ratio of VDD. The clocks gives insight into the constraints of the coupling cluster. For example, during the normal mode analysis the test clocks are supposed to be quiet. If there is no arrival windows, it reported as "No Arrival". Driving cell gives a idea relative strength of the victim and aggressor drivers. If the ports/pins have no driving cell it is reported as "No Lib Cell".

It also prints the other settings in design that could effect the crosstalk delay calculation, for example, logical correlation, composite aggressor setting, crosstalk delay analysis mode, and crosstalk delay effort level.

Attributes:

```
A - aggressor is Active
  C - aggressor is a composite aggressor
  E - aggressor is screened due to user Exclusion
  I - aggressor has Infinite arrival with respect to the victim
  L - aggressor is screened due to Logical correlation
  N - aggressor does Not overlap for the worst case alignment
  S - aggressor is screened for Small bumps
  U - aggressor/victim RC calculation is skipped.
  X - aggressor is screened due to aggressor eXclusion
```

Active aggressor is an aggressor that contributes to the worst case alignment scenario. It is reported as 'A'.

PrimeTime analyzes the composite effect of a group of weak aggressors (including filtered ones) for better quality of SI analysis. These aggressors are reported as 'C'.

You can exclude an aggressor from the analysis by using the *case_analysis*, *set_si_delay_analysis -exclude* or *set_coupling_separation* command. These aggressors are reported with attribute 'E'.

An aggressor has infinite window when it has no fixed timing relationship with the victim, for example, an aggressor clock is asynchronous to the victim clock. Also, if the victim or aggressor has the *set_si_delay_analysis -ignore_arrival* command, the aggressor gets the infinite window attribute 'I'.

When PrimeTime SI finds that the victim and aggressors cannot switch together due to the logical relationship (for example, the input and output net of a inverter, they cannot

switch in the same direction at the same time), PrimeTime SI chooses the combination of aggressors that can switch together logically and set them to active. The aggressors that cannot switch due to logical relationship are screened with attribute 'L'. The logical relationship could be identified only for buffer/inverter chains.

Due to multiple aggressors and their possible multiple arrival windows there could be multiple alignments which could cause crosstalk delta. PrimeTime SI identifies and reports the crosstalk delta for only the worst case alignment. If the aggressor does not contribute to the worst case alignment scenario, it is screened, and the attribute is set to 'N'.

The switching bumps that are smaller compared to the electrical filtering criteria, are screened. The attribute of these aggressors are set to 'S'. When composite aggressor feature is applied, the SI effect of screened aggressors are considered, and 'S' is replaced by 'C'.

If the aggressor or victim RC calculation is skipped, the attribute is set to 'U'. The common cause of the RC calculation skipping on a net is there is no cell arc driving the net(victim or aggressor) or the driving model could not be created.

The *set_si_aggressor_exclusion* command can set aggressor exclusion relationship among aggressors (for example, the specified aggressors cannot switch in the same direction at the same time), PrimeTime SI chooses the combination of aggressors that can switch together based on alignment and sets them to active. The aggressors that cannot switch due to aggressor exclusion relationship are screened with attribute 'X'.

Examples

The following example shows the output on a cell arc connected to an RC network. RC delay calculation is performed.

```
pt_shell> report_delay_calculation -thresholds -from U3/CK -to U3/Q
****************************************
Report : delay_calculation
 ...
****************************************

From pin: U3/CK
To pin:   U3/Q
Main Library Units:  1ns  0.001pF  1000kOhm

Library: 'slow'
Library Units:  1ns  1pF  1kOhm
Library Cell: 'SDFFX2'
arc sense: rising_edge
arc type:  cell

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage
 Temp.
```

```
 ------------------------------------------------------------------------
     from-pin  50     10->90    50     90->10    1.000     1.080
-40.0
       to-pin  50     10->90    50     90->10    1.000     1.080
-40.0

  RC network on pin 'U3/Q' :
  -----------------------------------------------------
  Number of elements = 2 Capacitances + 1 Resistances
  Total capacitance  = 0.094381 pF
  Total capacitance  = 94.380999 (in library unit)
  Total resistance   = 5.000000 Kohm

                             Rise        Fall
  ---------------------------------------------------------------------
  Input transition time  = 0.000000    0.000000  (in library unit)
  Effective capacitance  = 0.069497    0.061939  (in pF)
  Effective capacitance  = 69.496963    61.939442  (in library unit)
  Drive resistance       = 2.020383    1.252620  (in Kohm)
  Output transition time = 0.656019    0.389840  (in library unit)
  Cell delay             = 0.517020    0.356812  (in library unit)
```

The following example shows the output on a net arc. Wire load delay calculation is performed.

```
pt_shell> report_delay_calculation -min -thresholds -from U3/Q -to U4/A
****************************************
Report : delay_calculation
         -min
 ...
****************************************

From pin: U3/Q
To pin:   U4/A
Main Library Units:  1ns  0.001pF  1000kOhm

arc sense: unate
arc type:  net

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage
 Temp.

 ------------------------------------------------------------------------
     from-pin  50     10->90    50     90->10    1.000     1.080
 -40.0
       to-pin  50     10->90    50     90->10    1.000     1.080
 -40.0

RC delay calculation is being skipped because RC delay calculation was
not used for the driver arcs.
```

```
Balanced case tree
RC delay: (r_wire/load_count) * (c_pin + c_wire/load_count)
rise:     (0.005 / 1) * (4.381 + (90 / 1))
fall:     (0.005 / 1) * (4.381 + (90 / 1))


Rise delay = 0.471905
Fall delay = 0.471905
```

The following example shows the output on a cell arc. Table lookup into cell the library tables is performed.

```
pt_shell> report_delay_calculation -thresholds -from U1/A -to U1/Y
****************************************
Report : delay_calculation
 ...
****************************************

From pin: U1/A
To pin:   U1/Y
Main Library Units:  1ns  0.001pF  1000kOhm

Library: 'slow'
Library Units:  1ns  1pF  1kOhm
Library Cell: 'INVX2'
arc sense: negative_unate
arc type:  cell

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage
 Temp.

 ------------------------------------------------------------------------
     from-pin  50     10->90    50     90->10    1.000     1.080
 -40.0
       to-pin  50     10->90    50     90->10    1.000     1.080
 -40.0

RC delay calculation is being skipped because the driver from-pin is
 disabled.

Units:  1ns  1pF  1kOhm


 Rise Delay

   cell delay = 0.334919
     Table is indexed by
      (X) input_pin_transition = 0
      (Y) output_net_total_cap = 0.094381
     Relevant portion of lookup table:
                  (X)  0.0420      (X)  0.0660
  (Y)  0.0844      (Z)  0.3141      (Z)  0.3215
```

```
    (Y)  0.1706      (Z)  0.6050      (Z)  0.6125

       Z = A + B*X + C*Y + D*X*Y
       A =  0.0168           B =  0.3052
       C =  3.3703           D =  0.0362

    Z = 0.334919
    scaling result for operating conditions
    multiplying by 1 gives 0.334919

   Fall Delay

     cell delay = 0.215786
       Table is indexed by
        (X) input_pin_transition = 0
        (Y) output_net_total_cap = 0.094381
       Relevant portion of lookup table:
                    (X)  0.0420      (X)  0.0660
    (Y)  0.0844     (Z)  0.2071      (Z)  0.2145
    (Y)  0.1706     (Z)  0.3939      (Z)  0.4013

       Z = A + B*X + C*Y + D*X*Y
       A =  0.0115           B =  0.3079
       C =  2.1650           D =  0.0082

    Z = 0.215786
    scaling result for operating conditions
    multiplying by 1 gives 0.215786

    Cell Delay
    rise:  0.334919
    fall:  0.215786

  Rise delay = 0.334919
  Fall delay = 0.215786

  Units:  1ns  1pF  1kOhm

    Transition rise
     transition = 0.610174
       Table is indexed by
        (X) input_pin_transition = 0
        (Y) output_net_total_cap = 0.094381
       Relevant portion of lookup table:
                    (X)  0.0420      (X)  0.0660
    (Y)  0.0844     (Z)  0.5478      (Z)  0.5478
    (Y)  0.1706     (Z)  1.0854      (Z)  1.0854

       Z = A + B*X + C*Y + D*X*Y
       A =  0.0220           B =  0.0005
       C =  6.2318           D = -0.0058

    Z = 0.610174
```

```
  scaling result for operating conditions
  multiplying by 1 gives 0.610174


  Transition fall
   transition = 0.367266
     Table is indexed by
      (X) input_pin_transition = 0
      (Y) output_net_total_cap = 0.094381
     Relevant portion of lookup table:
                   (X)  0.0420      (X)  0.0660
  (Y)  0.0844      (Z)  0.3297      (Z)  0.3297
  (Y)  0.1706      (Z)  0.6536      (Z)  0.6536

     Z = A + B*X + C*Y + D*X*Y
     A =  0.0129           B = -0.0003
     C =  3.7542           D =  0.0029

  Z = 0.367266
  scaling result for operating conditions
  multiplying by 1 gives 0.367266


Rise transition = 0.610174
Fall transition = 0.367266
```

The following example shows the output on a net arc whose from-pin is a hierarchical pin. The delay of the net is assigned to the net arc whose from-pin is a leaf pin.

```
pt_shell> report_delay_calculation -from U1/clock_out -to U102/CP
****************************************
Report : delay_calculation
 ...
****************************************

RC delay calculation is being skipped because the load to-pin is not an
 input pin.

From pin: U1/clock_out
To pin:   U102/CP
Main Library Units:  1ns  1pF  1kOhm

arc sense: unate
arc type:  net

Net arc from a hierarchical pin.
Arc rise delay = 0 (assigned)
Arc fall delay = 0 (assigned)
To_pin rise transition time = 49.6413 (copied from from_pin)
To_pin fall transition time = 35.2787 (copied from from_pin)
```

Feedback

The following example shows the report for cell arc when the advanced mode of delay calculation is used. Advanced mode for driver model can be set using the *rc_driver_model_mode* variable. No additional option of *report_delay_calculation* is necessary to report details of the advanced mode of delay calculation.

```
pt_shell> report_delay_calculation -min -thresholds -from sig_in0 -to
 sig_in0 -nosplit
****************************************
Report : delay_calculation
         -min
 ...
****************************************

From port: sig_in0
To port:   sig_in0
Main Library Units:  1ns  1pF  1kOhm

Library: 'ports'
Library Units:  1ns  1pF  1kOhm
Library Cell: 'sig_in0'
arc sense: positive_unate
arc type:  cell

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage
 Temp.

 ------------------------------------------------------------------------
      from-pin  50     30->70    50     70->30    0.400     1.100
 125.0
        to-pin  50     30->70    50     70->30    0.400     1.100
 125.0

   RC network on pin 'sig_in0' :
   ----------------------------------------------------
   Number of elements = 8 Capacitances + 7 Resistances
   Total capacitance  = 0.003062 pF
   Total capacitance  = 0.003062 (in library unit)
   Total resistance   = 0.017983 Kohm

   Advanced driver-modeling used for rise and fall.

                             Rise        Fall
   ------------------------------------------------------------------
   Input transition time  =  0.100000    0.100000  (in library unit)
   Effective capacitance  =  0.002625    0.002800  (in pF)
   Effective capacitance  =  0.002625    0.002800  (in library unit)
   Output transition time =  0.060388    0.047040  (in library unit)
   Cell delay             =  0.050937    0.045125  (in library unit)
```

The following example shows the report for net arc when the advanced mode of delay calculation is used. You can set the advanced mode for the receiver model by using the

rc_receiver_model_mode variable. No additional option of the *report_delay_calculation* command is necessary to report details of the advanced mode of delay calculation. In addition to the existing information, the pin capacitances of the advanced receiver model and the transition values used to calculate the pin capacitances of the advanced receiver model are also reported.

```
pt_shell> report_delay_calculation -max -thresholds \\
            -from sig_in0 -to cell0/A -nosplit

From port: sig_in0
To pin:   cell0/A
Main Library Units:  1ns  1pF  1kOhm

arc sense: unate
arc type:  net

   Calculation  Rise   Rise      Fall   Fall      Slew      Rail
   Thresholds:  Delay  Slew      Delay  Slew      Derate    Voltage
 Temp.

 ------------------------------------------------------------------------
     from-pin  50     30->70    50     70->30    0.400     1.100
 125.0
       to-pin  50     30->70    50     70->30    0.400     1.100
 125.0

   RC network on pin 'sig_in0' :
   ----------------------------------------------------
   Number of elements = 8 Capacitances + 7 Resistances
   Total capacitance  = 0.003062 pF
   Total capacitance  = 0.003062 (in library unit)
   Total resistance   = 0.017983 Kohm

   Advanced receiver-modeling used for rise and fall.

   Advanced Receiver Model
                               Rise            Fall
   ------------------------------------------------------------------
   Receiver model capacitance  1 = 0.001966    0.001888  (in library
 unit)
   Receiver model capacitance  2 = 0.002328    0.002160  (in library
 unit)
   Receiver model transition   1 = 0.059192    0.037666  (in library
 unit)
   Receiver model transition   2 = 0.059192    0.037666  (in library
 unit)

                                Rise        Fall
   ------------------------------------------------------------------
   Net delay                =   0.000024    0.000064  (in library unit)
   Transition time          =   0.059192    0.037599  (in library unit)
   From_pin transition time =   0.059078    0.037666  (in library unit)
```

```
   To_pin transition time    =    0.059192     0.037599  (in library unit)
   Net slew degradation      =    0.000114    -0.000067  (in library unit)
```

The *define_scaling_lib_group* command allows you to define a group of libraries that PrimeTime interpolates between for voltage or temperature scaling. The following example shows the output on a net arc when scaling of the receiver model is done. The report indicates that scaling was done and also reports all the scaling libraries that were used for scaling the model.

```
pt_shell > report_delay_calculation -from I1/Z -to I2/B

From pin: I1/Z
To pin:   I2/B
Main Library Units:  1ns  0.001pF  1000kOhm

arc sense: unate
arc type:  net

   RC network on pin 'I1/Z' :
   -------------------------------------------------
   Number of elements = 2 Capacitances + 1 Resistances
   Total capacitance  = 0.815687 pF
   Total capacitance  = 815.686687 (in library unit)
   Total resistance   = 8.323139 Kohm

   Scaling library pin group used for rise and fall.
   Scaling libraries used for receiver model : tc300c_0.85  tc300c_1.05

                                Rise        Fall
   ---------------------------------------------------------------
   Net delay                 =    2.406542    2.836394  (in library unit)
   Transition time           =    5.312853    4.983530  (in library unit)
   From_pin transition time  =    0.804747    0.328441  (in library unit)
   To_pin transition time    =    5.312853    4.983530  (in library unit)
   Net slew degradation      =    4.508106    4.655089  (in library unit)
```

The following run shows the output on a net arc connected to a port. RC delay calculation with crosstalk is performed.

```
pt_shell> report_delay_calculation -crosstalk -from u2/Z -to o2
****************************************
Report : delay_calculation
Design : xtalk_test
****************************************

From pin: u2/Z
To port:   o2
Main Library Units:  1ns  1pF  1kOhm

arc sense: unate
arc type:  net
Annotated max rise net delta delay: 0 arc delay: 0.0151697
```

```
Annotated max fall net delta delay: 0 arc delay: 0.0153037

   RC network on pin 'u2/Z' :
   ----------------------------------------------------
   Number of elements = 5 Capacitances + 4 Resistances
   Total capacitance  = 0.300000 pF
   Total capacitance  = 0.300000 (in library unit)
   Total resistance   = 0.100000 Kohm

                             Rise        Fall
   ---------------------------------------------------------------------
   Net delay                = 0.015170    0.015304  (in library unit)
   Transition time          = 0.940437    0.542964  (in library unit)
   From_pin transition time = 0.940063    0.542372  (in library unit)
   To_pin transition time   = 0.940437    0.542964  (in library unit)
   Net slew degradation     = 0.000374    0.000592  (in library unit)

Annotated max rise delta transition: 0 pin transition: 0.940437
Annotated max fall delta transition: 0 pin transition: 0.542964
Reporting for Crosstalk:
Victim net name:                                    o2
Number of aggressors:                               1
Number of effective (non-filtered) aggressors:      1
Net is reselected:                                  true
Victim driver rail voltage(VDD):                    2.700000
si_xtalk_delay_analysis_mode:                       all_paths
si_analysis_logical_correlation_mode:               true
Crosstalk composite aggressor mode:                 disabled

Attributes:
  A - aggressor is Active
  C - aggressor is a composite aggressor
  E - aggressor is screened due to user Exclusion
  I - aggressor has Infinite arrival with respect to the victim
  L - aggressor is screened due to Logical correlation
  N - aggressor does Not overlap for the worst case alignment
  S - aggressor is screened for Small bumps
  U - aggressor/victim RC calculation is skipped
  X - aggressor is screened due to aggressor eXclusion

Victim is rising:
  Victim           Coupling    Driver        Clocks
   Net             Cap         Lib Cell
  -----------      ---------   ----------    ---------
  o2               0.200000    BF1T4_D        CLK2

  Aggressor        Coupling    Driver        Clocks        Attributes
 Switching Bump
   Net             Cap         Lib Cell
 (ratio of VDD)
 -------------     ---------   ---------     ---------     ------------
 -----------------
```

```
  o1                0.200000    BF1T4_D         CLK1           N
   -

Victim is falling:
  Victim            Coupling    Driver          Clocks
   Net              Cap         Lib Cell
  -----------       ---------   ----------      ---------
  o2                0.200000    BF1T4_D         CLK2

  Aggressor         Coupling    Driver          Clocks         Attributes
 Switching Bump
   Net              Cap         Lib Cell
 (ratio of VDD)
  -----------       ---------   ---------       ---------      ------------
 ----------------
  o1                0.200000    BF1T4_D         CLK1           N
   -
```

See Also

- define_scaling_lib_group
- get_timing_arcs
- report_lib
- report_lib_groups
- report_timing
- set_si_aggressor_exclusion
- rc_driver_model_mode
- rc_receiver_model_mode
- si_analysis_logical_correlation_mode
- si_xtalk_composite_aggr_mode
- si_xtalk_delay_analysis_mode

report_design

Shows information about the current design.

Syntax

status *report_design*

```
[-nosplit]
```

Feedback

Arguments

`-nosplit`

Prevents line splitting in the report when a field exceeds the column width.

Description

This command shows the following information about the current design:

- Operating conditions
- Wire load
- Design rules
- Timing ranges

Examples

The following example shows a design report.

```
pt_shell> report_design

****************************************
Report : design
Design : counter
 ...
****************************************

Design Attribute                  Value
------------------------------------------------------------------------------
----
Operating Conditions:
  operating_condition_max_name    WCCOM
  process_max                     1.50
  temperature_max                 70.00
  voltage_max                     4.75
  tree_type_max                   worst_case_tree

Wire Load:                        (use report_wire_load for more
 information)
  wire_load_mode                  top
  wire_load_model_max             --
  wire_load_selection_group_max   --
  wire_load_min_block_size        0

Design Rules:
  max_capacitance                 0.5
  min_capacitance                 --
  max_fanout                      5.6
  min_fanout                      --
  max_transition                  0.8
  min_transition                  --
```

```
  max_area                      --

Timing Ranges:
  fastest_factor                --
  slowest_factor                --
```

See Also

- report_clock
- report_port

---

report_design_mismatch

Reports all design mismatches found during linking.

Syntax

string *report_design_mismatch*

```
[-class cell | net | pin]
```

Arguments

`-class cell | net | pin`

For any pin that is found to have mismatches (such as instance and reference have different directions on the pin), only the associated objects (cells, nets, pins) that belong to this class are reported. The possible values are "cell", "net" and "pin".

Description

When the *link_allow_design_mismatch* variable is *true*, linking succeeds even when there are mismatches found. This allows you to gather as much useful information as possible even in the early stages of design. This command provides a report of mismatches for the current design.

Common causes of those mismatches include the following (listed in order of increasing priority):

1. A pin has different directions in instance and reference.
2. A pin of instance does not exist in reference.
3. A bus has different widths in instance and reference.

If a cell, pin, or net has a mismatch issue, the *is_design_mismatch* attribute returns *true* for the affected cell, pin, or net object. If the pin with a mismatch is on a bus, all other bits of the bus are also marked with the *is_design_mismatch* attribute.

Feedback

Note that if a cell or net has more than one mismatched issue, only the one with the highest priority is reported.

Examples

The following report example shows that the u1/Z and u2/Z pins have direction mismatches between instance and reference. The b net is connected to both u1/Z and u2/Z.

```
pt_shell> report_design_mismatch

****************************************
Report : report_design_mismatch
  ...
****************************************

pin                mismatch type
-----------------------------------------
u1/Z               Pin direction mismatch
u2/Z               Pin direction mismatch
-----------------------------------------
cell               mismatch type
-----------------------------------------
u1                 Pin direction mismatch
u2                 Pin direction mismatch
-----------------------------------------
net                mismatch type
-----------------------------------------
b                  Pin direction mismatch
```

See Also

- link_design
- link_allow_design_mismatch

report_design_variation

report_disable_pg_pins

Report the PG pins which are disabled using set_disable_pg_pins.

Syntax

string *report_disable_pg_pins*

[lib_cells]

Feedback

Data Types

lib_cells collection

Arguments

lib_cells

Specifies set of lib_cells on which disabled PG pins needs to be reported.

Description

The *report_disable_pg_pins* command reports the list of PG pins disabled using set_disable_pg_pins command. If lib cells are specified then, only the PG pins disabled for tat lib cell are reported. If no lib cells is specified all disabled PG pins are reported.

Examples

The following illustrates the report_disable_pg_pins command.

```
pt_shell> set_disable_pg_pins [get_lib_cell XYZ/A] -pg_pins {VDD}
1
pt_shell> report_disable_pg_pins
report_disable_pg_pins
****************************************
Report : report_lib_cell_disable_pg_pins
Design : Test
****************************************

LibCell                          Pin Name       Type
------------------------------------------------------------------
XYZ/A                            VDD            primary_power
```

See Also

- define_scaling_lib_group
- set_disable_pg_pins

---

report_disable_timing

Reports disabled timing arcs in the current design.

Syntax

string *report_disable_timing*

```
[-nosplit]
[cells_or_ports]
```

Feedback

Data Types

cells_or_ports collection

Arguments

-nosplit

Prevents line splitting to facilitate writing software to extract information from the report output. If you do not use this option, the report is shown in fixed-width columns.

cells_or_ports

Limits disabled arc reporting to the specified list of cells or ports. Provides the list or collection of cells or ports as an argument to the command.

Description

This command reports disabled timing arcs in the current design. Timing arcs can be disabled in several ways. You can disable a timing arc by using the *set_disable_timing* command. The following symbols are used to explain why a timing arc is disabled:

```
c : The propagation of case-analysis values

 C : The presence of conditional arcs (arcs that have a when statement
 defined in the library)

 d : The presence of default arcs (when
 the timing_disable_cond_default_arcs variable
 is set to true)

 f : Disabling of false net-arcs

 l : Loop breaking

 L : db inherited loop breaking

 m : Mode

 p : The propagation of constant values

 u : Arcs disabled by using the set_disable_timing command on cell arcs.

 U : Arcs disabled by using the set_disable_timing command on library
 cell arcs.
```

Examples

The following example shows that the timing arc of the n2_i cell from pin A to pin Z is disabled.

```
pt_shell> set_disable_timing {n2_i} -from A -to Z
pt_shell> report_disable_timing

****************************************
Report : disable_timing
 ...
****************************************

Flags :     c  case-analysis
            C  Conditional arc
            d  default conditional arc
            f  false net-arc
            l  loop breaking
            L  db inherited loop breaking
            m  mode
            p  propagated constant
            u  user-defined
            U  User-defined library arcs

Cell or Port         From    To      Sense           Flag  Reason
-------------------------------------------------------------------
n2_i                 A       Z       negative_unate  u

1
```

The following example specifies that the timing arc of the ff4 cell from pin CP to pin TE and TI are disabled as a result of a case-analysis constant of 0 propagated to pin TE of cell ff4. Note that the actual case value is set on another pin A and the propagation path to ff4/TE is inverting.

```
pt_shell> set_case_analysis 0 {p_in in0}
pt_shell> report_disable_timing

****************************************
Report : disable_timing
Design : middle
****************************************

Flags :     c  case-analysis
            C  Conditional arc
            d  default conditional arc
            f  false net-arc
            l  loop breaking
            L  db inherited loop breaking
            m  mode
            p  propagated constant
            u  user-defined
            U  User-defined library arcs

Cell or Port    From    To    Sense                 Flag  Reason
-------------------------------------------------------------------
o_reg4          CP      D     hold_clk_rise         c     D = 0
```

```
o_reg4          CP      D     setup_clk_rise          c     D = 0
o_reg4          CP      CD    recovery_rise_clk_rise  c     D = 0
p_out                                                 c     p_out = 1

1
```

To view disabled arcs in specific cells, provide a list or collection of the required cells as an argument to the command. To view disabled arcs for the o_reg4 instance, do the following:

```
pt_shell> report_disable_timing  [get_cells { o_reg4 }]

****************************************
Report : disable_timing
Design : middle
****************************************

Flags :     c  case-analysis
            C  Conditional arc
            d  default conditional arc
            f  false net-arc
            l  loop breaking
            L  db inherited loop breaking
            m  mode
            p  propagated constant
            u  user-defined
            U  User-defined library arcs

Cell or Port   From   To   Sense                  Flag    Reason
-------------------------------------------------------------------
o_reg4         CP     D    hold_clk_rise           c     D = 0
o_reg4         CP     D    setup_clk_rise          c     D = 0
o_reg4         CP     CD   recovery_rise_clk_rise  c     D = 0

1
```

See Also

- remove_disable_timing
- set_disable_timing

---

report_dominant_layer_in_path

Reports the layers with the most capacitance for specified nets.

Syntax

status *report_dominant_layer_in_path*

```
[-of_objects nets]
[-from node1 -to node2]
```

Feedback

Data Types

nets	list
node1	string
node2	string

Arguments

`-of_objects` *nets*

A list of one or more nets for which to report layer information. At least one net is required. Glob-style wildcards (*) are supported.

This option is mutually exclusive with the *-from* and *-to* options.

`-from` *node1*

Specifies a pin, port, or net internal node. The tool reports the layer information for paths between this node and the node specified in the *-to* option. Both nodes must both belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

`-to` *node2*

Specifies a pin, port, or net internal node. The tool reports the layer information for paths between this node and the node specified in the *-from* option. Both nodes must both belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

Description

This command reports the layers with the most capacitance for specified nets.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a dominant layer report.

```
pt_shell> report_dominant_layer_in_path -of_objects {SUM0 B0}
================================
List of nets in specified timing path:
net1: SUM0
net2: B0
Total number of nets in the timing path: 2

R dominant layer: poly
Total R on poly: 0.568534

C dominant layer: metal1
Total C on metal1: 0.055679
```

Feedback

report_drc_error

Writes out an error report of physical DRC errors from the specified error data. The *report_drc_error* command supports different reporting styles.

Syntax

status *report_drc_error*

```
-error_data    drc_error_data
[-error_type   drc_error_types]
[-of_objects   drc_error_objects]
[-report_type error_type | error_layer | detailed]
[-nosplit]
```

Data Types

```
drc_error_data     collection
drc_error_types    collection
drc_error_objects  collection
```

Arguments

`-error_data` *drc_error_data*

Specifies the error data for finding objects.

`-error_type` *drc_error_types*

Reports errors within the specified *drc_error_types*. An error is included in the output if it matches any one of the *drc_error_types*. Use the *get_drc_error_types* command to specify *drc_error_types*.

`-of_objects` *drc_error_objects*

Reports errors with the specified *drc_error_objects*. In this case, the *drc_error_objects* are error instances. Use the *get_drc_errors* command to specify *drc_error_objects*.

`-report_type error_type | error_layer | detailed`

Writes out the specified report type. Three types are supported: *error_type*, *error_layer*, and *detailed*. You can specify only one report type. By default, if no *report_type* is specified, the report type is *error_type*.

`-nosplit`

Write out the report without splitting long lines. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

Feedback

Description

The *report_drc_error* command prints a collection or summary of physical DRC errors in the specified error data that match certain criteria. The command returns true if any physical DRC errors match the criteria and successfully reported. If no objects match the criteria, the error status will be reported.

There are three types of report supported by the command: *error_type*, *error_layer*, and *detailed*. Use the *-report_type* option to specify the report type. The *error_type* report type writes out a summary of error information in type order. The *error_layer* report type writes out a summary of error information in layer order. The *detailed* report writes out each error instance information one-by-one. The information includes the ID, type, layer, number of errors in this type, description, status and bbox.

The *-error_type* option filters the errors by the specified types. You can specify more than one *error_type* here by enclosing each error type in curly braces: *{{$errType1} {$errType2}}*

The *-of_objects* option filters the errors by the specified error instances. Use the *get_drc_errors* command to specify the instances.

You can specify one or more filtering options to customize the report. If you specify *"Short"* and *"Net"* error type with *"METAL"* and *"METAL2"* layers, you can get errors with *"Short"* and *"METAL"*, with *"Short"* and *"METAL2"*, with *"Net"* and *"METAL"*, with *"Net"* and *"METAL2"*. If you specify the *-of_objects* option, the command writes out only the errors for the specified objects.

For long reports you do not want to write to the console, use the *redirect* command to write the output to a file.

Examples

The following example displays an error_type report of the errors with error type 'Net'.

```
prompt> set type [get_drc_error_types -error_data $data {Net}]
prompt> report_drc_error -error_data $data -error_type $type -report_type
 error_type

****************************************
Report : Report drc errors
Design : test
Data   : zroute.err
Type   : error_type
Version: O-2018.06-SP2-BETA
Date   : Mon Jul  9 15:26:16 2018
****************************************

ErrorSet                                        Total   Visible     Ignored
 Fixed
```

```
------------------------------------------------------------------------
-------
Design : test                                54       3           0       0
      Data : zroute.err                      54       3           0       0
            Net                              3        3           0       0
                METAL2 (18)                  1        1           0       0
                METAL3 (28)                  1        1           0       0
                METAL4 (31)                  1        1           0       0
1
```

See Also

- redirect
- get_drc_errors
- get_drc_error_types

---

report_driver_model

Displays the driver model for a library cell timing arc used to drive annotated parasitics.

Syntax

integer *report_driver_model*

```
-lib_cell lib_cell
-from_pin from_pin
-to_pin to_pin
-rise_slew rise_slew
-fall_slew fall_slew
-capacitance capacitance
```

Data Types

```
lib_cell            string
from_pin            string
to_pin              string
rise_slew           float
fall_slew           float
capacitance         double
```

Arguments

`-lib_cell` *lib_cell*

Specifies the name of the library cell for which the driver model is computed. Specify the name in the *library_name/cell_name* format.

`-from_pin` *from_pin*

Specifies the startpoint of a timing arc through the *lib_cell* value.

`-to_pin` *`to_pin`*

Specifies the endpoint of a timing arc through the *lib_cell* value.

`-rise_slew` *`rise_slew`*

Specifies the rise time in library units on the *from_pin* value.

`-fall_slew` *`fall_slew`*

Specifies the fall time in library units on the *from_pin* value.

`-capacitance` *`capacitance`*

Specifies the load in library units on the *to_pin* value.

Description

The *report_driver_model* command provides both the library data and the resulting driver model parameters for the specified library arc and conditions. This information is useful for validating library data or PrimeTime driver models used in delay calculation with annotated parasitics.

For each timing arc between the specified pins, the delay, slew, and sensitivities (the changes in delay and slew resulting from a small change in load capacitance) are displayed. This information can be compared to simulation results using the same input slew and output load to measure library interpolation errors. If the current design is in min-max mode, data for both operating conditions is displayed.

Slews are displayed without derating (they are shown as the time between the trip-points.).

In addition to the library data, the driver model parameters used in delay calculation with annotated parasitics are displayed. Currently, PrimeTime supports only the simplified driver model (SDM) with three parameters: rd (the drive resistance through which a linear voltage ramp is applied), tz (the ramp start time relative to the arc input threshold time), and dt (the full-swing ramp duration).

The driver model is constructed to match the library delay, slew, and sensitivity for the specified input slew and output load.

Problematic library data can prevent the construction of a driver model altogether. In this case, the driver model parameters displays as zero, and the check shows FAIL. The most common reason for this error is that the library data not indicating an increasing delay or slew for increasing output capacitance; this indicates a nonpositive drive resistance. Another common reason is incorrect trip-point settings.

Examples

The following report shows the output for the driver model for a library cell timing arc.

```
pt_shell> report_driver_model \\
  -lib_cell [get_lib_cell -of_objects I0] \\
```

Feedback

```
  -from_pin A -to_pin Y \\
  -rise_slew 0.0384 -fall_slew 0.0384 -capacitance 0.115194


****************************************
Report : driver_model
Driver : core/inv27:A-->Y
Version: 2000.11
Date   : Thu Aug  7 14:43:08 2000
****************************************

Accuracy Settings:              Rise  Fall
    slew_lower_threshold     =  30%   30%
    slew_upper_threshold     =  70%   70%
    input_threshold          =  50%   50%
    output_threshold         =  50%   50%
    slew_derate_from_library =       1

Library Inputs:  (in library units)
    Rising  input slew      =  0.038400
    Falling input slew      =  0.038400
    Output load capacitance =  0.115194

Driver model for sense 'negative-unate':
    (max)        Rise      Fall
    ---------------------------------------------------
    load     =  0.115194  0.115194  (pF)
    delay    =  0.071811  0.044181  (ns)
    slew     =  0.043008  0.026270  (ns without derate)
    d-load   =  0.001152  0.001152  (pF)
    d-delay  =  0.000392  0.000264  (ns)
    d-slew   =  0.000332  0.000160  (ns without derate)
    rd       =  0.406621  0.249222  (kohm)
    tz       = -0.000323  0.000139  (ns)
    dt       =  0.070798  0.043157  (ns)
    error    =  0.853807  0.950170  (%)
    check    =    pass       pass
```

See Also

- report_delay_calculation
- rc_driver_model_mode

---

report_eco_library_cells

Reports alternative library cells considered for PrimeTime ECO commands.

Syntax

string *report_eco_library_cells*

```
[-pattern_priority pattern_list]
[-attribute attribute_name]
[-show_others]
[-current_library]
[-nosplit]
[-power_attribute power_attribute_name]
```

Data Types

pattern_list	list
attribute_name	string
power_attribute_name	string

Arguments

`-pattern_priority` *pattern_list*

Specifies a list of library cell patterns starting from the highest to lowest priority.

`-attribute` *attribute_name*

Specifies an attribute name that can be used to match patterns specified by the *-pattern_priority* option. When you specify this option, PrimeTime uses the value of the attribute instead of library cell names for pattern matching.

`-show_others`

Shows library cells that do not match the specified pattern. This option must be used together with the *-pattern_priority* option.

`-current_library`

Shows alternative library cells within the same library.

`-nosplit`

Prevents line splitting when column overflows to facilitate writing script to extract information from the report output.

`-power_attribute` *power_attribute_name*

Specifies a power attribute set on library cells. If you specify this option, the attribute values are reported in a separate column.

Description

This command reports the alternative library cells of all cells in a design. This command must be used by workers for multi-scenario analysis flow within PrimeTime.

Examples

The following example reports alternative library cells:

```
pt_shell> report_eco_library_cells
```

```
****************************************
Report : eco_library_cells
Design : design
 ...
****************************************

Alternative library cells:

  Attributes:
      u - dont_use or pt_dont_use
      d - dont_touch

Group  Library_Cell                                  Area Attributes
--------------------------------------------------------------------
[   0] LIBRARY/BUFX1_LVT                             2.12
       LIBRARY/BUFX2_LVT                             2.64
       LIBRARY/BUFX4_LVT                             4.76 d
       LIBRARY/BUFX8_LVT                             7.41 u d
[   1] LIBRARY/DFFPQX1_LVT                           9.52
       LIBRARY/DFFPQX2_LVT                           9.52
       LIBRARY/DFFPQX4_LVT                          12.70
```

The following example reports alternative library cells with patterns HVT and LVT:

```
pt_shell> report_eco_library_cells -pattern_priority {HVT LVT}

****************************************
Report : eco_library_cells
        -pattern_priority { HVT LVT }
Design : design
 ...
****************************************

Alternative library cells:

  Attributes:
      u - dont_use or pt_dont_use
      d - dont_touch

Group  Library_Cell                                  Area Attributes
--------------------------------------------------------------------
[   0] LIBRARY/BUFX1_HVT                             2.12
       LIBRARY/BUFX1_LVT                             2.12
[   1] LIBRARY/BUFX2_HVT                             2.64
       LIBRARY/BUFX2_LVT                             2.64
[   2] LIBRARY/DFFX1_HVT                             9.52 u
       LIBRARY/DFFX1_LVT                             9.52
```

The following example reports alternative library cells with power attribute:

```
pt_shell> report_eco_library_cells -power_attribute power_attr
****************************************
Report : eco_library_cells
```

Feedback

```
        -power_attribute power_attr
Design : design
 ...
**************************************

Alternative library cells:

  Attributes:
      u - dont_use or pt_dont_use
      d - dont_touch

Group Library_Cell                                  Area Power_Attr
 Attributes
-------------------------------------------------------------------------------
-----
[  0] LIBRARY_H/INV_X0_HVT                          1.27   13773.83
      LIBRARY_H/INV_X1_HVT                          1.27   25491.85
      LIBRARY_R/INV_X0_RVT                          1.27   26041.65
      LIBRARY_R/INV_X1_RVT                          1.27   48138.68
      LIBRARY_H/INV_X2_HVT                          1.52   53638.71
      LIBRARY_R/INV_X2_RVT                          1.52  101260.30
      LIBRARY_H/INV_X4_HVT                          2.03  117975.20
      LIBRARY_R/INV_X4_RVT                          2.03  222585.09 u
```

See Also

- fix_eco_leakage
- fix_eco_power
- eco_alternative_cell_attribute_restrictions

---

report_eco_mim_instances

Reports ECO MIM instances in the current design.

Syntax

string *report_eco_mim_instances*

```
[-summary]
```

Arguments

`-summary`

Generates a summary report that omits the "Sources" column and limits the number of reported instances to 10 per MIM group.

Feedback

Description

This command reports the sets of multiply instantiated module (MIM) instances that can be modified identically by ECO operations. For each ECO MIM group, the report shows the number of instances, the source of the MIM group, the Verilog module name, and the instance names.

The "source" of a MIM group is the type of action that created the MIM group:

- *Parasitics*: By reading a single parasitic data file for multiple instances
- *Physical*: By module grouping in the physical data files (DEF or NDM)
- *User-Defined*: By explicit MIM grouping using the *set_eco_options* command with the *-mim_group* option
- *Netlist Editing*: By uniquification resulting from netlist editing commands like *insert_buffer*, *size_cell*, and *create_net*

Examples

The following is an example of an ECO MIM instances report.

```
pt_shell> report_eco_mim_instances
  ...
ID    # Insts    Sources         Module          Instances
-------------------------------------------------------------------
0     12         Parasitics      MIM_BLOCK       { block_I10 block_I11
 block_I12 block_I13 }

1     2          Physical        MIM_BLOCK_0     { block_I0 block_I1 }

2     2          User-Defined    MIM_BLOCK_1     { block_I2 block_I3 }
```

See Also

- read_parasitics
- set_eco_options

---

report_eco_options

Reports options specified by the *set_eco_options* command.

Syntax

string *report_eco_options*

Arguments

None.

Feedback

Description

The *report_eco_options* command reports all the ECO options specified by the *set_eco_options* command.

Distributed Multi-Scenario Analysis

To execute the *report_eco_options* command in all scenarios, use the *remote_execute* command. For example:

```
remote_execute -verbose {
  report_eco_options
}
```

Note: You cannot execute the *report_eco_options* command in the manager process.

Examples

The following example reports the ECO options previously specified by the *set_eco_options* command.

```
pt_shell> report_eco_options
****************************************
Report : eco_options
Design : DESIGN
****************************************

Technology LEF files
------------------------------
tech.lef

Cell LEF files
------------------------------
top.lef
block1.lef
block2.lef
block3.lef

DEF files                                   Physical constraint files
------------------------------              --------------------------
top.def                                     top_va.tcl
block1.def                                  block1_va.tcl
block2.def                                  block2_va.tcl
block3.def                                  block3_va.tcl

Physical ICC2 lib                           Physical ICC2 blocks
------------------------------              --------------------------
ndm_lib
                                           block4


User Filler Cell Names
------------------------------
```

Feedback

```
FILL
FILLCAP1

Log file                                        Log format
------------------------------                  --------------------------

ECO margins
----------------------------------------
drc_setup_margin              -INFINITY
drc_hold_margin               -INFINITY
timing_setup_margin               0.000
timing_hold_margin                0.000
power_setup_margin                0.000
power_hold_margin                 0.000
```

See Also

- reset_eco_options
- set_eco_options

report_eco_scenarios

report_eco_wire

Reports ECO wire optimization changes in the design.

Syntax

```
status report_eco_wire
[-verbose]
[net_list]
```

Data Types

```
net_list          list
```

Arguments

`-verbose`

Displays verbose ECO wire optimization information.

net_list

Lists of nets to report the ECO wire optimization changes.

Feedback

Description

This command reports ECO wire optimization changes performed by *fix_eco_wire* or *set_routing_rule* commands. Changes by *set_routing_rule* will be reported by "manual" method.

Examples

The following example reports the ECO wire optimization change to a net named *n43*.

```
prompt> report_eco_wire {n43}
```

See Also

- fix_eco_wire
- reset_eco_wire

---

report_est_power_savings

Generates estimated power saving reports.

Syntax

string *report_est_power_savings*

```
[-type  power_saving_type]
[-min_reg_width  register_width]
[-max_clk_effi   clock_efficiency]
[-min_pwr_savings  power_savings]
[-self_gating_type  gating_type]
[-csv csv_file]
[-prob_based_gating]
[-report_nl_names]
[object_list]
```

Data Types

power_saving_type	string
register_width	integer
clock_efficiency	integer
power_savings	float
gating_type	string
csv_file	string
object_list	list

Feedback

Arguments

`-type` *`power_saving_type`*

Use this option to specify the power saving technique to be used on the registers for estimating the power savings data on them. Currently only "self_gating" value can be specified.

`-min_reg_width` *`register_width`*

Use this option to filter the bundled registers so that the bundled registers having width greater than or equal to "register_width" will be considered for power saving techniques. Default value of *-min_reg_width* is 8.

`-max_clk_effi` *`clock_efficiency`*

Use this this option to filter the registers so that the registers with a domain clock frequency less than or equal to "clock_efficiency" will be considered for power saving techniques. Default value of *-max_clk_effi* is 80.

`-min_pwr_savings` *`power_savings`*

Use this option to report only those registers whose total power saving, after using the power saving techniques, is more than or equal to the "power_savings" value. Default value of *-min_pwr_savings* is 0.0.

`-self_gating_type` *`gating_type`*

Use this option to specify the explicit gate type to be used when using the "self_gating" type power saving technique. Supported gate types are XOR, OR and NAND. This option is only useful when the user does not want to use the *-prob_based_gating* option. Default gate type used is "XOR" when "self_gating" type is used.

`-csv` *`csv_file`*

Use this option to specify the path for the csv file for writing the power savings report in a comma seperated value format. A csv file can be viewed in a spreadsheet reader. With this option the report will not be printed on the shell.

`-prob_based_gating`

Use this option to automatically determine the gate type for "self_gating" based on the static probability of D and Q Pins. This option cannnot be used with *-self_gating_type* option. The possible gate types are selected from OR, NAND or XOR, based on the values of static probability of the registers involved.

`-report_nl_names`

Use this option to report the netlist(gate-level) name of the register. By default, the RTL name or merged RTL name, in case of bus register, is reported for the registers.

object_list

Use this option to apply the power savings techniques on the list of registers. Options *-min_reg_width*, *-max_clk_effi* and *-min_pwr_savings* cannot be supported with this option. Currently only the gate-level register name can be specified.

Description

Use the *report_est_power_savings* command to generate the power savings report after applying the power savings techniques on the register/bus registers in the design. The two important fields that the command prints are the name of registers and the potential power savings that could happen using the given power savings technique. The other fields are : frequency of clock, if the register is gated or not, the register gate efficiency percentage, current power, final power and the gate-type used for the self-gating method type.

Currently only the "self_gating" power saving method is supported.

To use the command, you should first apply the update_power and update_metrics command or apply the compute_metrics command.

Self_Gating power saving technique

In traditional clock gating technique, the clock is gated based on the enable signal. But if the enable signal is not optimal, then the Q/CP ratio of the register could be low. For such scenarios, a self gating technique is generally used involving an XOR gate to improve the Q/CP ratio and hence making clock gating effective for a given register. With XOR self-gating the enable signal is generated by performing an XOR of D and Q pins of the sequential cell, thus the enable will be high only when D dffers from Q.

Register Name in report

The command prints the RTL register name or the merged RTL names in case of bus registers by default. If mapping information does not exist for some registers the gate level name would get printed instead.

To print the gate-level register names instead of the RTL names use the option -report_nl_names.

Examples

The following example shows a list of bundled registers having a width >=32, where the power could be saved by employing XOR self-gating. The registers may have either a gated clock or not.

```
pt_shell> report_est_power_savings -type self_gating -min_reg_width 32
****************************************
Report : report_est_power_savings
Design : mephisto_core
Version: R-2020.09-SP5-1-DEV-20210805
Date   : Thu Aug  5 22:21:26 2021
```

```
****************************************

--------------------------------------------------------------------------------
-------------------------------------------------------------------
Register                              Gated          Clock Pin      Current
       Current        Final            Predicted      Gating
Name                                  (Y/N)          Frequency       Clock
 Gating    Power          Power           Clock Power    Type
                                                       (MHz)
 Efficiency (%) (W)             (W)             Savings (W)
--------------------------------------------------------------------------------
-------------------------------------------------------------------
mep_dp_0/ra_0/Q0_r[0:31]              Gated          333.3          12.8
       3.055e-05      1.461e-05      1.594e-05      XOR
mep_pc_0/n_mac_r[0:51]                Gated          333.3          0.0
       2.903e-05      1.734e-05      1.169e-05      XOR
mep_dp_0/ram_1/ram_0/bist/i_q_r[0:31]
                                      Not Gated      333.3          0.0
       1.166e-05      2.754e-07      1.139e-05      XOR
mep_dp_0/ram_0/ram_0/bist/i_q_r[0:31]
                                      Not Gated      333.3          0.0
       1.166e-05      2.754e-07      1.139e-05      XOR
mep_pc_0/n_mac_r5[0:31]               Gated          333.3          0.0
       1.112e-05      4.517e-07      1.067e-05      XOR
mep_pc_0/n_mac_r7[0:31]               Gated          333.3          0.0
       1.086e-05      7.787e-07      1.008e-05      XOR
mep_pc_0/cfg_seq_ptr_r3[0:31]         Gated          333.3          0.0
       1.126e-05      1.312e-06      9.944e-06      XOR
mep_pc_0/cfg_seq_ptr_r7[0:31]         Gated          333.3          0.0
       1.126e-05      1.317e-06      9.939e-06      XOR
mep_pc_0/cfg_seq_ptr_r0[0:31]         Gated          333.3          5.4
       1.005e-05      3.664e-07      9.681e-06      XOR
mep_pc_0/n_mac_r3[0:31]               Gated          333.3          6.8
       1.002e-05      3.951e-07      9.626e-06      XOR
mep_pc_0/n_ram1_r4[0:31]              Gated          333.3          0.0
       9.864e-06      3.839e-07      9.48e-06       XOR
mep_pc_0/n_ram1_r2[0:31]              Gated          333.3          0.1
       9.625e-06      3.97e-07       9.228e-06      XOR
mep_pc_0/n_ram1_r7[0:31]              Gated          333.3          0.0
       9.572e-06      3.907e-07      9.182e-06      XOR
mep_pc_0/n_ram1_r5[0:31]              Gated          333.3          0.0
       9.308e-06      3.833e-07      8.925e-06      XOR
mep_pc_0/ram_0/bist_1/i_q_r[0:31]     Not Gated      333.3          0.0
       9.136e-06      2.421e-07      8.894e-06      XOR
mep_pc_0/ram_0/bist_0/i_q_r[0:31]     Not Gated      333.3          0.0
       9.136e-06      2.421e-07      8.894e-06      XOR
mep_pc_0/cache_r1[0:31]               Gated          333.3          7.5
       9.074e-06      3.843e-07      8.69e-06       XOR
mep_dp_0/mac_1/st2_Mi1[0:31]          Gated          333.3          69.4
       8.297e-06      7.496e-06      8.005e-07      XOR
--------------------------------------------------------------------------------
-------------------------------------------------------------------
1
```

Feedback

The following example shows a list of bundled register having width >=32, where the power could be saved by employing XOR self-gating technique. The registers may have gated clock or not. The use of the option *report_nl_names* has resulted in printing the netlist names of bundled register (gate-level).

```
pt_shell> report_est_power_savings -type self_gating -min_reg_width 32
 -report_nl_names
****************************************
Report : report_est_power_savings
Design : mephisto_core
Version: R-2020.09-SP5-1-DEV-20210805
Date   : Thu Aug  5 22:21:33 2021
****************************************

--------------------------------------------------------------------------------
-----------------------------------------------------------------
Register                              Gated          Clock Pin      Current
       Current        Final            Predicted      Gating
Name                                  (Y/N)          Frequency       Clock
 Gating   Power          Power          Clock Power    Type
                                                      (MHz)
 Efficiency (%) (W)             (W)             Savings (W)
--------------------------------------------------------------------------------
-----------------------------------------------------------------
mep_dp_0/ra_0/Q0_r_reg[0:31]          Gated          333.3          12.8
       3.055e-05      1.461e-05       1.594e-05      XOR
mep_pc_0/n_mac_r_reg[0:51]            Gated          333.3          0.0
       2.903e-05      1.734e-05       1.169e-05      XOR
mep_dp_0/ram_0/ram_0/bist/i_q_r_reg[0:31]
                                      Not Gated      333.3          0.0
       1.166e-05      2.754e-07       1.139e-05      XOR
mep_dp_0/ram_1/ram_0/bist/i_q_r_reg[0:31]
                                      Not Gated      333.3          0.0
       1.166e-05      2.754e-07       1.139e-05      XOR
mep_pc_0/n_mac_r5_reg[0:31]           Gated          333.3          0.0
       1.112e-05      4.517e-07       1.067e-05      XOR
mep_pc_0/n_mac_r7_reg[0:31]           Gated          333.3          0.0
       1.086e-05      7.787e-07       1.008e-05      XOR
mep_pc_0/cfg_seq_ptr_r3_reg[0:31]     Gated          333.3          0.0
       1.126e-05      1.312e-06       9.944e-06      XOR
mep_pc_0/cfg_seq_ptr_r7_reg[0:31]     Gated          333.3          0.0
       1.126e-05      1.317e-06       9.939e-06      XOR
mep_pc_0/cfg_seq_ptr_r0_reg[0:31]     Gated          333.3          5.4
       1.005e-05      3.664e-07       9.681e-06      XOR
mep_pc_0/n_mac_r3_reg[0:31]           Gated          333.3          6.8
       1.002e-05      3.951e-07       9.626e-06      XOR
mep_pc_0/n_ram1_r4_reg[0:31]          Gated          333.3          0.0
       9.864e-06      3.839e-07       9.48e-06       XOR
mep_pc_0/n_ram1_r2_reg[0:31]          Gated          333.3          0.1
       9.625e-06      3.97e-07        9.228e-06      XOR
mep_pc_0/n_ram1_r7_reg[0:31]          Gated          333.3          0.0
       9.572e-06      3.907e-07       9.182e-06      XOR
```

```
mep_pc_0/n_ram1_r5_reg[0:31]       Gated          333.3          0.0
       9.308e-06      3.833e-07      8.925e-06      XOR
mep_pc_0/ram_0/bist_1/i_q_r_reg[0:31]
                                   Not Gated      333.3          0.0
       9.136e-06      2.421e-07      8.894e-06      XOR
mep_pc_0/ram_0/bist_0/i_q_r_reg[0:31]
                                   Not Gated      333.3          0.0
       9.136e-06      2.421e-07      8.894e-06      XOR
mep_pc_0/cache_r1_reg[0:31]        Gated          333.3          7.5
       9.074e-06      3.843e-07      8.69e-06       XOR
mep_dp_0/mac_1/st2_Mi1_reg[0:31]   Gated          333.3          69.4
       8.297e-06      7.496e-06      8.005e-07      XOR
--------------------------------------------------------------------------------
--------------------------------------------------------------------
1
```

The following example shows a list of bundled registers having width >=32, where the power could be saved using self-gating involving the use of either a XOR, OR, or NAND gate, depending on the values of the static probabilities of register pins. For bus registers, the user can observe "MULTIPLE" gating type which means that some registers in the bus use, different gates for the power savings.

```
pt_shell> report_est_power_savings -type self_gating -min_reg_width 32
 -prob_based_gating
****************************************
Report : report_est_power_savings
Design : mephisto_core
Version: R-2020.09-SP5-1-DEV-20210805
Date   : Thu Aug  5 22:21:41 2021
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------
Register                           Gated          Clock Pin      Current
       Current        Final          Predicted      Gating
Name                               (Y/N)          Frequency       Clock
 Gating   Power          Power          Clock Power    Type
                                                    (MHz)
 Efficiency (%) (W)             (W)             Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------
mep_dp_0/adr_sr_0/buf0[0:63]       Gated          333.3          4.6
       2.301e-05      4.952e-07      2.252e-05      MULTIPLE
mep_pc_0/n_mac_r[0:51]             Gated          333.3          0.0
       2.903e-05      9.298e-06      1.973e-05      MULTIPLE
mep_dp_0/ra_0/Q0_r[0:31]           Gated          333.3          12.8
       3.055e-05      1.461e-05      1.594e-05      XOR
mep_pc_0/cfg_seq_ptr_r3[0:31]      Gated          333.3          0.0
       1.126e-05      1.312e-06      9.944e-06      XOR
mep_pc_0/cfg_seq_ptr_r7[0:31]      Gated          333.3          0.0
       1.126e-05      1.317e-06      9.939e-06      XOR
```

```
mep_dp_0/ra_0/D0_r[0:31]            Gated          333.3          8.0
       1.522e-05      7.782e-06      7.434e-06      XOR
mep_pc_0/cfg_seq_ptr_r5[0:31]       Gated          333.3          47.8
       6.074e-06      1.726e-06      4.348e-06      XOR
mep_dp_0/ram_0/data_out[0:31]       Gated          333.3          76.1
       8.994e-06      6.068e-06      2.926e-06      XOR
mep_dp_0/mac_1/st2_Mj1[0:31]        Gated          333.3          69.4
       8.931e-06      7.49e-06       1.441e-06      MULTIPLE
--------------------------------------------------------------------------------
--------------------------------------------------------------------
1
```

The following example shows a list of bundled registers having a width >=32 and a clock efficiency <=50, where the power could be saved by employing XOR self-gating.

```
pt_shell> report_est_power_savings -type self_gating -min_reg_width 32
 -max_clk_effi 50
****************************************
Report : report_est_power_savings
Design : mephisto_core
Version: R-2020.09-SP5-1-DEV-20210805
Date   : Thu Aug  5 22:21:47 2021
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------
Register                            Gated          Clock Pin      Current
       Current        Final          Predicted      Gating
Name                                (Y/N)          Frequency       Clock
 Gating   Power          Power          Clock Power    Type
                                                    (MHz)
 Efficiency (%) (W)             (W)             Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------
mep_dp_0/ra_0/Q0_r[0:31]            Gated          333.3          12.8
       3.055e-05      1.461e-05      1.594e-05      XOR
mep_pc_0/n_mac_r[0:51]              Gated          333.3          0.0
       2.903e-05      1.734e-05      1.169e-05      XOR
mep_dp_0/ram_0/ram_0/bist/i_q_r[0:31]
                                    Not Gated      333.3          0.0
       1.166e-05      2.754e-07      1.139e-05      XOR
mep_dp_0/ram_1/ram_0/bist/i_q_r[0:31]
                                    Not Gated      333.3          0.0
       1.166e-05      2.754e-07      1.139e-05      XOR
mep_pc_0/n_mac_r5[0:31]             Gated          333.3          0.0
       1.112e-05      4.517e-07      1.067e-05      XOR
mep_pc_0/n_mac_r7[0:31]             Gated          333.3          0.0
       1.086e-05      7.787e-07      1.008e-05      XOR
mep_pc_0/cfg_seq_ptr_r3[0:31]       Gated          333.3          0.0
       1.126e-05      1.312e-06      9.944e-06      XOR
mep_pc_0/cfg_seq_ptr_r7[0:31]       Gated          333.3          0.0
       1.126e-05      1.317e-06      9.939e-06      XOR
```

```
mep_pc_0/cfg_seq_ptr_r0[0:31]      Gated          333.3          5.4
       1.005e-05      3.664e-07      9.681e-06      XOR
mep_pc_0/n_mac_r3[0:31]            Gated          333.3          6.8
       1.002e-05      3.951e-07      9.626e-06      XOR
mep_pc_0/n_ram1_r4[0:31]           Gated          333.3          0.0
       9.864e-06      3.839e-07      9.48e-06       XOR
mep_pc_0/n_ram1_r2[0:31]           Gated          333.3          0.1
       9.625e-06      3.97e-07       9.228e-06      XOR
mep_pc_0/n_ram1_r7[0:31]           Gated          333.3          0.0
       9.572e-06      3.907e-07      9.182e-06      XOR
mep_pc_0/n_ram1_r5[0:31]           Gated          333.3          0.0
       9.308e-06      3.833e-07      8.925e-06      XOR
mep_pc_0/ram_0/bist_0/i_q_r[0:31]  Not Gated      333.3          0.0
       9.136e-06      2.421e-07      8.894e-06      XOR
mep_pc_0/ram_0/bist_1/i_q_r[0:31]  Not Gated      333.3          0.0
       9.136e-06      2.421e-07      8.894e-06      XOR
mep_pc_0/cache_r1[0:31]            Gated          333.3          7.5
       9.074e-06      3.843e-07      8.69e-06       XOR
mep_pc_0/cfg_seq_ptr_r5[0:31]      Gated          333.3          47.8
       6.074e-06      1.726e-06      4.348e-06      XOR
--------------------------------------------------------------------------------
--------------------------------------------------------------------
1
```

The following example shows a list of bundled registers having width >=32, clock efficiency <=50 and where power saved is than 9.681e-06W. The power has been saved by employing XOR self-gating.

```
pt_shell> report_est_power_savings -type self_gating -min_reg_width 32
 -max_clk_effi 50 -min_pwr_savings 9.681e-06
****************************************
Report : report_est_power_savings
Design : mephisto_core
Version: R-2020.09-SP5-1-DEV-20210805
Date   : Thu Aug  5 22:21:53 2021
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------
Register                           Gated          Clock Pin      Current
       Current        Final          Predicted      Gating
Name                               (Y/N)          Frequency       Clock
 Gating   Power          Power          Clock Power    Type
                                                    (MHz)
 Efficiency (%) (W)             (W)             Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------
mep_dp_0/ra_0/Q0_r[0:31]           Gated          333.3          12.8
       3.055e-05      1.461e-05      1.594e-05      XOR
mep_pc_0/n_mac_r[0:51]             Gated          333.3          0.0
       2.903e-05      1.734e-05      1.169e-05      XOR
mep_dp_0/ram_1/ram_0/bist/i_q_r[0:31]
```

```
                                    Not Gated      333.3          0.0
       1.166e-05      2.754e-07      1.139e-05      XOR
mep_dp_0/ram_0/ram_0/bist/i_q_r[0:31]
                                    Not Gated      333.3          0.0
       1.166e-05      2.754e-07      1.139e-05      XOR
mep_pc_0/n_mac_r5[0:31]             Gated          333.3          0.0
       1.112e-05      4.517e-07      1.067e-05      XOR
mep_pc_0/n_mac_r7[0:31]             Gated          333.3          0.0
       1.086e-05      7.787e-07      1.008e-05      XOR
mep_pc_0/cfg_seq_ptr_r3[0:31]       Gated          333.3          0.0
       1.126e-05      1.312e-06      9.944e-06      XOR
mep_pc_0/cfg_seq_ptr_r7[0:31]       Gated          333.3          0.0
       1.126e-05      1.317e-06      9.939e-06      XOR
--------------------------------------------------------------------------------
--------------------------------------------------------------------
1
```

The following example shows a list of bundled registers, where power has been saved by employing XOR self-gating. The user must give the gate-level register name when specifying the list of registers to be used.

```
pt_shell> report_est_power_savings -type self_gating [get_cells
 mep_dp_0/ra_0/Q0_r[*]]
****************************************
Report : report_est_power_savings
Design : mephisto_core
Version: R-2020.09-SP5-1-DEV-20210805
Date   : Thu Aug  5 22:21:53 2021
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------
Register                            Gated          Clock Pin      Current
       Current        Final          Predicted      Gating
Name                                (Y/N)          Frequency       Clock
 Gating    Power          Power          Clock Power    Type
                                                      (MHz)
 Efficiency (%) (W)             (W)             Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------
mep_dp_0/ra_0/Q0_r[0:31]            Gated          333.3          12.8
       3.055e-05      1.461e-05      1.594e-05      XOR
--------------------------------------------------------------------------------
--------------------------------------------------------------------
1
```

See Also

- compute_metrics
- update_power

- update_metrics
- report_power
- report_rtl_metrics

---

report_etm_arc

Reports the data and clock paths traversed while extracting a particular timing arc.

Syntax

string *report_etm_arc*

```
[-from from_object
   | -rise_from rise_from_object
   | -fall_from fall_from_object]
[-to to_object
   | -rise_to rise_to_object]
   | -fall_to fall_to_object]
[-arc_type arc_type]
[-include path_type_list]
[-context_borrow]
[-latch_level levels]
[-library_cell]
[-etm_report file_name]
[-netlist_report file_name]
[-significant_digits digits]
[-variation]
[-sms_scenarios sms_scenarios_list]
```

Data Types

```
from_object          list
rise_from_object     list
fall_from_object     list
to_object            list
rise_to_object       list
fall_to_object       list
arc_type             list
path_type_list       list
levels               integer
file_name            string
digits               integer
sms_scenarios_list   collection
```

Feedback

Arguments

`-from` *`from_object`*

Specifies a port or clock from which the arc of interest originates. The sense at the starting point can be either rising or falling. Substitute one of the following valid values for *from_object*: *-fall_from*, *-from*, or *-rise_from*.

`-rise_from` *`rise_from_object`*

Specifies a port or clock from which the arc of interest originates. The sense at the startpoint must be rising.

`-fall_from` *`fall_from_object`*

Specifies a port or clock from which the arc of interest originates. The sense at the startpoint must be falling.

`-to` *`to_object`*

Specifies a port or clock at which the arc of interest terminates. The sense at the endpoint can be either rising or falling. Substitute one of the following valid values for *to_object*: *-fall_to*.,*-rise_to*, or *-to*.

`-rise_to` *`rise_to_object`*

Specifies a port or clock at which the arc of interest terminates. The sense at the endpoint must be rising.

`-fall_to` *`fall_to_object`*

Specifies a port or clock at which the arc of interest terminates. The sense at the endpoint must be falling.

`-arc_type` *`arc_type`*

Specifies the type of arc reported between the start and endpoint.

Only a single type can be specified. Valid values are: *setup*, *hold*, *clock_gating_setup*, *clock_gating_hold*, *recovery*, *removal*, *non_seq_setup*, *non_seq_hold*, *max_combo_delay*, *min_combo_delay*, *max_seq_delay*, and *min_seq_delay*.

`-include` *`path_type_list`*

Specifies what types of path information should be reported. The reporting behaviors are:

- default reporting (*-include* not specified)
  - Extracted data path
- *-include {clock_path}*

    - Extracted clock and data paths
- *-include {netlist_path}*
    - Extracted data path
    - Full-netlist data path
- *-include {clock_path netlist_path}*
    - Extracted clock and data paths
    - Full-netlist clock and data paths

where "extracted" refers to the path as viewed by the *extract_model* command, and "full-netlist" refers to the path as viewed by the PrimeTime timing engine in the current full-netlist analysis.

Extracted paths are reported in an "ETM Report" section, while full-netlist reports are reported in a "Netlist Report" section.

`-context_borrow`

Specifies that PrimeTime determines the latches on an interface that borrow, based on input port arrival times and clock definitions specified at the time of model extraction. This option is one of three that specify the model extraction environment in which the debugging is done: *-context_borrow*, *-latch_level*, and *-library_cell*. The *-context_borrow* and *-latch_level* options are mutually exclusive.

`-latch_level` *`levels`*

Specifies the number of levels of latch borrowing that occur at the interface of a design. This option is one of three that specify the model extraction environment in which the debugging is done: *-context_borrow*, *-latch_level*, and *-library_cell*. The *-context_borrow* and *-latch_level* options are mutually exclusive.

`-library_cell`

Specifies that the model is extracted as a library cell. This option eliminates boundary nets, and the boundary parasitics become part of the model. This option is one of three that specify the model extraction environment in which the debugging is done: *-context_borrow*, *-latch_level*, and *-library_cell*.

`-etm_report` *`file_name`*

Specifies that the ETM report should be written to the specified file.

`-netlist_report` *`file_name`*

Specifies that the full-netlist report should be written to the specified file.

`-significant_digits` *digits*

Specifies the number of digits after the decimal point is displayed for time values in the generated report. Substitute one of the following valid values for *digits*: an integer from 0 to 13. The default is 2. You can use this option to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

`-variation`

Specifies the report to include parametric on-chip variation (POCV) information in columns labeled "Mean" and "Sensit". POCV analysis is enabled by setting the *timing_pocvm_enable_analysis* variable to *true*.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only etm arc data compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *report_etm_arc* command reports details about the timing paths used to extract a particular timing arc.

It is typically used to debug a discrepancy between an ETM and a netlist reported by the *write_interface_timing* and *compare_interface_timing* model validation commands.

This command is run in the full-netlist analysis. It internally runs the ETM extraction engine on a boundary timing path of interest, then reports the path as viewed by extraction. The extracted path is reported in the usual timing path format, as a sequence of pins with arc delays and cumulative arrivals.

By default, only the data path portion of the extracted timing arc is reported. The *-include* option can be used to include additional path information in the output, such as also including the clock path, or also including the path as viewed by the current full-netlist timing analysis.

To compare ETM paths to full-netlist paths, it is recommended to use the *-include netlist_path* option of this command instead of the *report_timing* command for the following reasons:

- The *report_timing* command might not report the correct path. For example, *report_timing* would report the worst-slack path to an output port, whereas ETM extraction (and thus this command) use the worst-delay path instead.
- Some types of launch and capture paths might be difficult to report with the *report_timing* command, such as for some constraint and check arcs.

Feedback

You must use the same model extraction options and environment variables when generating a model as when debugging the model. This ensures that the same arc is extracted during model extraction and debugging.

Therefore, the values of environment variables used by the *extract_model* command are also valid for the *report_etm_arc* command. The *-context_borrow*, *-latch_level*, and *-library_cell* options specify how to perform extraction.

Examples

The following example generates a *report_etm_arc* command from a discrepancy shown by model validation. If the *compare_interface_timing* command yields a failure as:

```
RBUS_RnotW(r)  ADC_SCLK_IN(r)  setup  17.18  18.24  1.06  FAIL

pt_shell> report_etm_arc -rise_from RBUS_RnotW \\
            -rise_to [get_clock ADC_SCLK_IN] -arc_type setup \\
            -include {clock_path netlist_path}
```

The following example generates a *report_etm_arc* command for a clock to output path. From the following model validation discrepancy:

```
ADC_SCLK_IN(r) ADC_LRCLK_OUT(f)  FALL  2.04  2.71  0.67  FAIL

pt_shell> report_etm_arc \\
            -rise_from [get_clock ADC_SCLK_IN] \\
            -fall_to ADC_LRCLK_OUT -arc_type min_seq_delay \\
            -include {clock_path netlist_path}
```

See Also

- compare_interface_timing
- extract_model
- report_timing
- write_interface_timing

---

report_exceptions

Reports timing exceptions.

Syntax

status *report_exceptions*

```
[-from from_list]
[-rise_from rise_from_list]
[-fall_from fall_from_list]
[-through through_list]
```

Feedback

```
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list]
[-rise_to rise_to_list]
[-fall_to fall_to_list]
[-ignored]
[-nosplit]
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
through_list	list
rise_through_list	list
fall_through_list	list
to_list	list
rise_to_list	list
fall_to_list	list

Arguments

`-from` *from_list*

Specifies a list of clocks, ports, cells, and pins in the current design. The report includes only paths that start at the objects in the *from_list* value. Using this option limits the report to information that was set using a path-based command (for example, the *set_multicycle_path* command) with the *-from* option. The *-from*, *-rise_from*, and *-fall_from* options are mutually exclusive.

`-rise_from` *rise_from_list*

Same as the *-from* option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can only use one of the *-from*, *-rise_from*, and *-fall_from* options.

`-fall_from` *fall_from_list*

Same as the *-from* option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can only use one of the *-from*, *-rise_from*, and *-fall_from* options.

`-through` *through_list*

Specifies a list of pins or ports. The report includes only paths that go through the pins or ports on the *through_list* value. Using this option limits the report to information that was set using a path-based command (for example, the

set_multicycle_path command) with the *-through* option. You can use this option more than one time in the same command.

`-rise_through` *`rise_through_list`*

Specifies a list of pins or ports (the same as the *-through* option) except that the paths must have a rising transition at the through points. You can use this option more than one time in the same command.

`-fall_through` *`fall_through_list`*

Specifies a list of pins or ports (the same as the *-through* option) except that the paths must have a falling transition at the through points. You can use this option more than one time in the same command.

`-to` *`to_list`*

Specifies a list of clocks, ports, cells, and pins in the current design. The report is to include only paths that end at the objects in the *to_list* objects. Using this option limits the report to information that was set using a path-based command (for example, the *set_multicycle_path* command) with the *-to* option. The *-to*, *-rise_to*, and *-fall_to* options are mutually exclusive.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can only use one of the *-to*, *-rise_to*, and *-fall_to* options.

`-fall_to` *`fall_to_list`*

Same as the *-to* option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can only use one of the *-to*, *-rise_to*, and *-fall_to* options.

`-ignored`

Indicates that the report lists path timing exceptions that are set on the current design, but are completely ignored. For example, a false path might be specified from port A to port Z1, but if there is no timing path between those points, the path is ignored. Use the *-from* or *-to* options to limit the report to certain paths. Ignored exceptions on objects are also included in the report; for example, an exception placed by the *set_max_time_borrow* command on a cell other than a level-sensitive latch.

`-nosplit`

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

Description

The *report_exceptions* command reports information about timing exceptions for the current design.

PrimeTime accepts a timing exception if it alters the constraints of the path. If path constraints are different in the absence of a timing exception, the exception alters the path constraint.

The *report_exceptions* command attempts to identify reasons an exception is ignored, fully or partially, and classifies these into one of the following: invalid startpoints, invalid endpoints, non-existent paths, or overridden paths. This is reported as flags in the Ignored column. When the *-ignored* option is used, the reported flags indicates why an exception is fully ignored. Otherwise, the reported flags indicates why a subset of the exception specified paths were ignored.

By default (without the *-ignored* option), *report_exceptions* displays dominant exceptions. An exception only appears in this section if a path exists for which this exception applies. That is, a path exists that

- Matches the exception's topological and setup/hold, rise/fall criteria
- Has a valid startpoint and endpoint
- Satisfies no higher-priority exception

If no such path exists, the exception is reported in the ignored report with explanatory reasons why the exception was ignored. Even if the exception is dominant, some parts of its specification might be unsatisfiable. Consequently, each of the following reason flags can appear beside an exception whether the exception is dominant or ignored.

- p: invalid path

  At least one of the pins in through or to sets is not reachable from any pin in its predecessor (-from or -through) pin set, the specified exception is deemed to have an invalid path specification.

- f: invalid start point

  At least one of the pins in from set does not launch a signal. This flag is shown only if a from set is specified in the exception.

- t: invalid endpoint

At least one of the pins in -to set is not an endpoint in a timing analysis sense. This flag is shown only if a to set is specified in the exception.

- o: overridden exception

  In this case the exception specification is met by some timing path but another exception is applicable to the path (due to a higher exception precedence or more constraining value).

If a path is unconstrained (that is, if it has a required time of infinity), timing exceptions do not change the path constraints. Applying a timing exception to an unconstrained path does not change the path constraint. A path is unconstrained if it has an infinite required time.

To remove timing exceptions from specified paths, use the *reset_path* command. The *reset_design* command removes all attributes from the design, including timing exceptions.

The number of timing exception objects reported in the *from*, *through*, or *to* sets are limited by the value of the *collection_result_display_limit* variable. As in collection result display, setting the variable to a negative value would print the complete set. Any nonnegative value would truncate any set when the size is greater than the value; the truncated output is designated by an ellipsis. Note that for negative settings, the behavior matches object collection query echoing the entire object set. It is recommended to avoid negative settings when reporting large numbers of exceptions with sizable object sets as this can adversely affect the performance of the *report_exceptions* command.

The max_time_borrowed report is generated for global reporting only and not when any of the *-from*, *-through*, or *-to* options are specified with the command.

Examples

The following example lists all timing exceptions set on the design.

```
pt_shell> report_exceptions

****************************************
Report : exceptions
Design : counter
 ...
****************************************

Reasons:
    f - invalid startpoint(s)
    t - invalid endpoint(s)
    p - non-existent paths
    o - overridden paths

From          To         Setup          Hold                Ignored
```

```
--------------------------------------------------------------------------------
----
RESET           *           FALSE           FALSE
*               CO          max=4.5         min_rise=2,min_fall=2.5
ffa             ffb         cycles=2        -
```

The following example shows rise/fall qualified timing exceptions set on the design.

```
pt_shell> report_exceptions

From            To          Setup           Hold                  Ignored
--------------------------------------------------------------------------------
----
RESET           *           FALSE           FALSE
*               CO(r)       max=4.5         min_rise=2,min_fall=2.5
ffa(r)          ffb(f)      cycles=2        -
```

The following example lists all ignored timing exceptions set on the design.

```
pt_shell> report_exceptions -ignored

****************************************
Report : exceptions
         -ignored
Design : counter
 ...
****************************************

Reasons:
    f - invalid startpoint(s)
    t - invalid endpoint(s)
    p - non-existent paths
    o - overridden paths

From            To          Setup           Hold                  Ignored
--------------------------------------------------------------------------------
----
QA              *           max=10          -                     f
*               A           cycles=2        -                     t
*               B           FALSE           FALSE                 t


Object                   Type   Attributes
--------------------------------------------------------------------------------
----
CLK                      clock  max_time_borrow=2.3
```

The following example lists some user-entered compressed exceptions.

```
pt_shell> report_exceptions

****************************************
Report : exceptions
```

```
Design : counter
 ...
****************************************

Reasons:
    f - invalid startpoint(s)
    t - invalid endpoint(s)
    p - non-existent paths
    o - overridden paths

From          Through        To          Setup          Hold
 Ignored
------------------------------------------------------------------------
-------
{ ffa/CP ffb/CP }
              *              { ffa/D ffb/D ffc/D }     -
                                max=2                          p
```

See Also

- current_design
- set_false_path
- set_max_delay
- set_max_time_borrow
- set_min_delay
- set_multicycle_path
- collection_result_display_limit

---

report_glitch

Reports glitch source information.

Syntax

string *report_glitch*

```
[-rtl]
[-gate]
[-source]
[-fanout_cone]
[-id source_id]
[-display_source_instances]
[-verbose]
[-debug_source]
```

Data Types

id number

Arguments

-rtl

This option specifies to report glitch source information for rtl design.

-gate

This option specifies to report glitch source information for gate level design.

-source

This option reports all the glitch sources in the design as per power impacted by either RTL line or Gate instance. Each source is represented by a unique id. Additional information about the glitch source and its fanout are also reported like the hier cell of the source, type of leaf cells present in the glitch fanout with their count etc.

-fanout_cone

With the *-id* option, this option lists all the leaf level instances that are part of the glitch fanout represented by cone id. This option can only be used after *report_glitch -source* has been run.

-id *source_id*

This option specifies the glitch source id for which report is to be generated. It must be used along with *-fanout_cone* option.

-display_source_instances

This option lists RTL source code cross-reference information and reports all the leaf level source instances that are part of the glitch source for each source ID. This option can only be used with *report_glitch -source* command. It also reports other information like fanout cone glitch power and fanout count which are reported by *report_glitch -source* command.

-verbose

This option provides additional information about the leaf level source instances that are part of glitch source for each source ID. It reports glitch power in fanout cone of each leaf instance, parent instance name and parent cross-reference information of each glitch source. This option can only be used with *report_glitch -source -display_source_instances* command.

-debug_source

This option provides detailed debug information about glitch source for a given source ID. It reports arrival time and path details of glitch source inputs.

It reports logic depth, RTL cross-reference and RTL constructs in min and max paths of fanin cone of glitch source. This option can only be used with *report_glitch -rtl -fanout_cone -id* command.

Description

This command provides various information helpful in debugging glitch sources in the design. The sources are ranked based on glitch power consumption.

To use the command, you should first apply the *update_power* and *update_metrics* command or apply the *compute_metrics* command.

Examples

The following example shows the default report of *-source -rtl* and helps user to report all glitch sources in design along with RTL cross-reference, fanout count and glitch power:

```
pwr_shell> report_glitch -source -rtl

****************************************
Report : report_glitch
      -rtl
      -source
Design : top
****************************************

------------------------------------------------------------
   Printing glitch sources as per power impacted by RTL line
------------------------------------------------------------

ID - 1
Source - /rtl_power/INPUT/data/alu.v:30
Power - 1.420e-07  Count- 137

Source Hier Instances (Power)
i_alu(1.420e-07)

Module (InstCount) (Directly Connected) (Glitch Power)

------------------------

ID - 2
Source - /rtl_power/INPUT/data/alu.v:28
Power - 1.261e-07  Count- 77

Source Hier Instances (Power)
i_alu(1.261e-07)

Module (InstCount) (Directly Connected) (Glitch Power)

------------------------
1
```

```
To report the source instances for each glitch source, user can use
 following option:

pwr_shell> report_glitch -rtl -source -display_source_instances
****************************************
Report : report_glitch
      -rtl
      -source
Design : top
****************************************

------------------------------------------------------------
   Printing glitch sources as per power impacted by RTL line
------------------------------------------------------------

ID - 1
Source - /rtl_power/INPUT/data/alu.v:30
Power - 1.420e-07  Count- 137

Source Instances
i_alu/ctmi_184
i_alu/ctmi_118
i_alu/ctmi_186
i_alu/ctmi_147
i_alu/ctmi_149
i_alu/ctmi_103
i_alu/ctmi_109
i_alu/ctmi_105
i_alu/ctmi_116
i_alu/SGI23_350
i_alu/ctmi_108

Source Hier Instances (Power)
i_alu(1.420e-07)

Module (InstCount) (Directly Connected) (Glitch Power)

------------------------

ID - 2
Source - /rtl_power/INPUT/data/alu.v:28
Power - 1.261e-07  Count- 77

Source Instances
i_alu/A232
i_alu/SGI42_279
i_alu/SGI32_301
i_alu/SGI39_299
i_alu/SGI36_313
i_alu/SGI38_303
i_alu/A231
i_alu/A233
```

```
i_alu/SGI35_297
i_alu/ctmi_86

Source Hier Instances (Power)
i_alu(1.261e-07)

Module (InstCount) (Directly Connected) (Glitch Power)

------------------------
1

To report the source instances for each glitch source, user can use
 following option:

pwr_shell> report_glitch -rtl -source -display_source_instances -verbose
****************************************
Report : report_glitch
      -rtl
      -source
Design : top
Version: S-2021.06-SP5-2-DEV-20221117
Date   : Thu Nov 17 20:04:54 2022
****************************************


------------------------------------------------------------
   Printing glitch sources as per power impacted by RTL line
------------------------------------------------------------

ID - 1
Source - /rtl_power/INPUT/data/alu.v:30
Power - 1.420e-07  Count- 137

Source Instances

Glitch power in fanout of source 1.1 : i_alu/ctmi_184: 1.757691e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.2 : i_alu/ctmi_118: 1.446629e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.3 : i_alu/ctmi_186: 1.345342e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.4 : i_alu/ctmi_147: 1.293425e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.5 : i_alu/ctmi_149: 1.142442e-09
Parent (i_alu) :
```

```
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.6 : i_alu/ctmi_103: 1.671803e-07
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.7 : i_alu/ctmi_109: 9.341133e-10
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.8 : i_alu/ctmi_105: 5.369701e-11
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.9 : i_alu/ctmi_116: 4.121372e-11
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.10 : i_alu/SGI23_350: 6.114439e-11
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 1.11 : i_alu/ctmi_108: 2.625455e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Source Hier Instances (Power)
i_alu(1.420e-07)

Module (InstCount) (Directly Connected) (Glitch Power)

Family (InstCount)
HDBLVT(6)
HDBSVT(131)

Celltype List (Count)
   HDBSVT16_AOAI211_1(1)
   HDBLVT16_OA211_1(1)
   HDBSVT16_OAI21_V1T_1P5(2)
   HDBLVT16_OAI211_1(1)
   HDBSVT16_NR2B_1P25(1)
   HDBSVT16_NR2_1P25(1)
   HDBSVT16_NR2_1(8)
   HDBSVT16_NR2B_1(2)
   HDBLVT16_OAOI211_1(1)
   HDBSVT16_INV_1(13)
   HDBSVT16_OAOI211_1(2)
   HDBSVT16_MAJI3_T_2(1)
   HDBSVT16_AOI21_1(4)
   HDBLVT16_OAI21_1(1)
   HDBSVT16_MUXI2_1(7)
   HDBSVT16_AOA211_2(1)
   HDBSVT16_ADDF_V1_1P25(1)
```

```
    HDBSVT16_AOI21_TV1_2(4)
    HDBSVT16_OA2BB2_1(2)
    HDBSVT16_OA21_1(1)
    HDBLVT16_AO21_1(1)
    HDBSVT16_NR2_T_1P25(1)
    HDBSVT16_ND2_1(10)
    HDBSVT16_OAI2111_1(1)
    HDBSVT16_AOI21_3(1)
    HDBSVT16_AO2BB2_G_1(2)
    HDBSVT16_AOI21B_1(1)
    HDBSVT16_AOI2222_V2_1(5)
    HDBSVT16_FSDPQM4_DLPY2_1(2)
    HDBSVT16_EN2_V2_1(8)
    HDBSVT16_NR3B_UG_1(2)
    HDBSVT16_ADDF_V1_1P5(8)
    HDBSVT16_NR2B_4(1)
    HDBSVT16_AN2_D_4(1)
    HDBSVT16_OAOI211_V2_1(6)
    HDBSVT16_ND2_2(1)
    HDBSVT16_MAJI3_1(1)
    HDBSVT16_ND3B_1(2)
    HDBSVT16_OAI21_V1T_1P25(1)
    HDBSVT16_MUX2_MA_1(1)
    HDBSVT16_ND2B_V1U_1(2)
    HDBSVT16_MAJI3B_1(1)
    HDBSVT16_EO2_V2_1(4)
    HDBSVT16_OAI311_1(1)
    HDBSVT16_ND2_3(1)
    HDBSVT16_AOI211_1(3)
    HDBSVT16_AO21B_1(1)
    HDBSVT16_OAI22_1(1)
    HDBLVT16_MAJI3B_1(1)
    HDBSVT16_EN2_V2_2(1)
    HDBSVT16_OAI221_1(3)
    HDBSVT16_EO2_F_2(1)
    HDBSVT16_OAI21_1(5)
    HDBSVT16_NR2_3(1)

------------------------

ID - 2
Source - /rtl_power/INPUT/data/alu.v:28
Power - 1.261e-07  Count- 77

Source Instances

Glitch power in fanout of source 2.1 : i_alu/A232: 1.324753e-08
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.2 : i_alu/SGI42_279: 5.812019e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17
```

```
Glitch power in fanout of source 2.3 : i_alu/SGI32_301: 3.263290e-08
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.4 : i_alu/SGI39_299: 2.051231e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.5 : i_alu/SGI36_313: 5.116811e-10
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.6 : i_alu/SGI38_303: 4.669590e-08
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.7 : i_alu/A231: 2.869761e-08
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.8 : i_alu/A233: 3.636524e-09
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.9 : i_alu/SGI35_297: 4.744656e-11
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Glitch power in fanout of source 2.10 : i_alu/ctmi_86: 2.468666e-08
Parent (i_alu) :
/rtl_power/INPUT/data/top.v:17

Source Hier Instances (Power)
i_alu(1.261e-07)

Module (InstCount) (Directly Connected) (Glitch Power)

Family (InstCount)
HDBSVT(77)

Celltype List (Count)
   HDBSVT16_OAOI211_1(1)
   HDBSVT16_NR4BB_1(1)
   HDBSVT16_OAI211_1(1)
   HDBSVT16_ADDF_V1_1(11)
   HDBSVT16_AO221_1(10)
   HDBSVT16_OAOI211_V2_1(4)
   HDBSVT16_FSDPQM4_DLPY2_1(2)
   HDBSVT16_AOI221_1(1)
   HDBSVT16_EO2_V2_1(2)
   HDBSVT16_EN2_V2_1(3)
   HDBSVT16_EN3_1(1)
```

```
   HDBSVT16_AO222_1P5(1)
   HDBSVT16_AOI22_1(1)
   HDBSVT16_OAI311_1(1)
   HDBSVT16_NR2_1(8)
   HDBSVT16_AOI31_1(1)
   HDBSVT16_OAI221_1(5)
   HDBSVT16_AOI211_1(4)
   HDBSVT16_ND2_1(5)
   HDBSVT16_OAI2111_1(1)
   HDBSVT16_AOI21_1(2)
   HDBSVT16_AOA211_2(1)
   HDBSVT16_EO3_1(1)
   HDBSVT16_ADDH_1(1)
   HDBSVT16_AOI222_1(1)
   HDBSVT16_AOI2222_V2_1(5)
   HDBSVT16_OA211_1(2)

------------------------
1

To report the fanout cone instances for a given glitch source and source
 ID, user can use following option:

pwr_shell> report_glitch -rtl -id 33 -fanout_cone
**************************************
Report : report_glitch
      -rtl
      -fanout_cone
Design : top
**************************************

ID - 33

Source - i_alu/ctmi_1572

Cell Instances -
 i_alu/ctmi_1572
 i_alu/ALU_Result_reg[7:4]
 i_alu/ctmi_1544
 i_alu/ctmi_1570
 i_alu/ctmi_1693
 i_alu/SGI27_262
 i_alu/SGI27_269
 i_alu/SGI30_265
 i_alu/ctmi_1571
 i_alu/ctmi_196
 i_alu/ALU_Result_reg[3:0]

1


To print detailed debugging information about a given glitch source ID,
 user can use following option:
```

Feedback

```
pwr_shell> report_glitch -fanout_cone -id 11 -rtl -debug_source
****************************************
Report : report_glitch
      -rtl
      -fanout_cone
Design : top
****************************************

ID - 11

Source - /rtl_power/INPUT/data/alu.v:19

Cell Instances -

Source Instance: i_alu/ctmi_223
Arrival window for pin i_alu/ctmi_223/A1 @ rising {Clock: io_clk, period:
 2, Min: 0.475387, Max: 0.481991, Diff: 0.006604 }
Arrival window for pin i_alu/ctmi_223/A2 @ rising {Clock: io_clk, period:
 2, Min: 0.491713, Max: 0.513360, Diff: 0.021647 }
Arrival window for pin i_alu/ctmi_223/B1 @ rising {Clock: io_clk, period:
 2, Min: 0.449772, Max: 0.449772, Diff: 0.000000 }
Arrival window for pin i_alu/ctmi_223/B2 @ rising {Clock: io_clk, period:
 2, Min: 0.486384, Max: 0.501608, Diff: 0.015224 }

Max path of depth: 4 for pin : i_alu/ctmi_223/A2
Min path of depth: 1 for pin : i_alu/ctmi_223/B1

Path to Glitch Source Pin: i_alu/ctmi_223/A2
Cell           Libcell        Logic                          RTL
                 RTL
                                Depth                          Construct
                 Source
-------------------------------------------------------------------------
-------------------------------------------------------------------------
----
i_alu/ctmi_1548 HDBSVT16_INV_1  4
  /rtl_power/INPUT/data/alu.v:5
i_alu/ctmi_1550 HDBSVT16_INV_1  3
  /rtl_power/INPUT/data/alu.v:5
i_alu/ctmi_1551 HDBSVT16_INV_1  3
  /rtl_power/INPUT/data/alu.v:5
i_alu/ctmi_1547 HDBSVT16_NR2_1  3
  /rtl_power/INPUT/data/alu.v:22
i_alu/ctmi_1557 HDBSVT16_ND2_1  2
  /rtl_power/INPUT/data/alu.v:22
i_alu/ctmi_1558 HDBSVT16_INV_1  2
  /rtl_power/INPUT/data/alu.v:22
i_alu/ctmi_1556 HDBSVT16_NR2_1  1
  /rtl_power/INPUT/data/alu.v:22
i_alu/ctmi_223  HDBSVT16_OA2BB2_1 0
  /rtl_power/INPUT/data/alu.v:19
```

```
Path to Glitch Source Pin: i_alu/ctmi_223/B1
Cell            Libcell        Logic                          RTL
                  RTL
                               Depth                          Construct
                  Source
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
----
i_alu/ctmi_1593 HDBSVT16_INV_1  1
  /rtl_power/INPUT/data/alu.v:4
i_alu/ctmi_223  HDBSVT16_OA2BB2_1 0
  /rtl_power/INPUT/data/alu.v:19


Fanout Cone Instances -
 i_alu/ALU_Result_reg[3:0]
 i_alu/ctmi_196
 i_alu/SGI29_271
 i_alu/SGI27_269

1
```

See Also

- compute_metrics
- update_power
- update_metrics

---

report_global_slack

Displays slack of specified pins or ports.

Syntax

string *report_global_slack*

```
[-significant_digits digits]
[-nosplit]
[-min]
[-max]
[-rise]
[-fall]
[-sms_scenarios sms_scenarios_list]
port_pin_list
```

Data Types

```
digits               integer
port_pin_list        list
sms_scenarios_list   collection
```

Feedback

Arguments

`-significant_digits` *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default is 2. Use this option if you want to override the default.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. *-nosplit* prevents line splitting and facilitates writing software to extract information from the report output.

`-min`

Displays the minimum slack.

`-max`

Displays the maximum slack.

`-rise`

Displays rise slack.

`-fall`

Displays fall slack.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins and ports in the current design. Pins of hierarchical cells are not reported.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only global slack information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *report_global_slack* command displays the slack of specified pins and ports. By default, all pins except those of hierarchical cells and ports of the design are reported. The slack shown in the report for a given pin corresponds to the worst slack among all paths traversing that pin. If a pin is both a timing startpoint and endpoint then its slack will be the worst among all paths either starting and ending at the pin.

To report slacks throughout the current design arrival totals and slacks must be available on all pins, not just at endpoints. To achieve this, set the

Feedback

timing_save_pin_arrival_and_slack variable to *true* before issuing the *report_global_slack* command. If the *timing_save_pin_arrival_and_slack* variable has not been set to *true*. the command is set it to *true* and updates the design timing before the command executes.

The default of the variable is *false*. If you intend to use this command, it is recommended that you set the *timing_save_pin_arrival_and_slack* variable to *true* before the first timing update, thus preventing the cost of an additional timing update.

The *-max* and *-min* options are mutually exclusive. The *-rise* and *-fall* options are also mutually exclusive. You can choose any combination from the *-max* or *-min* option and *-rise* or *-fall* option to report a particular slack value.

- If you specify only the *-max* option, but not the *-rise* or *-fall* option, the tool reports both the maximum rise and maximum fall slack values.
- If you specify only the *-min* option, but not the *-rise* or *-fall* option, the tool reports both the minimum rise and minimum fall slack values.
- If you specify only the *-rise* option, but not the *-max* or *-min* option, the tool reports both the maximum rise and minimum rise slack values.
- If you specify only the *-fall* option, but not the *-max* or *-min* option, the tool reports both the maximum fall and minimum fall slack values.
- If you specify neither the *-rise* and *-fall* options nor the *-max* and *-min* options, the tool reports the maximum rise, maximum fall, minimum rise, and minimum fall slack values.

If slack attribute is not available at a pin or port, the tool prints *.

Note that this command expects that the design has already been timed with the *timing_save_pin_arrival_and_slack* variable set to *true*. If the *timing_save_pin_arrival_and_slack* variable is set to its default of *false*, the *report_global_slack* command can potentially take longer during runtime.

Examples

The following example generates a report of all slack violations in the current design.

```
pt_shell> report_global_slack -significant_digits 4 -nosplit
****************************************
Report : report_global_slack
 ...
****************************************

Max_Rise      Max_Fall      Min_Rise      Min_Fall   Point
-------------------------------------------------------------
-1.4869       -1.6330        1.5669        1.7130   ff2/D
-1.4869       -1.6330        1.5669        1.7130   in2/Z
-1.6330       -1.4869        1.7130        1.5669   in2/A
-1.6330       -1.4869        1.7130        1.5669   in1/Z
```

Feedback

See Also

- report_timing
- report_default_significant_digits
- timing_save_pin_arrival_and_slack
- pin_attributes

---

report_global_timing

Reports a top-level summary of the design setup and hold timing: WNS, TNS, and number of violations.

Syntax

status *report_global_timing*

```
[-delay_type max | min]
[-group group_list]
[-include include_list]
[-format format_spec]
[-output file_name]
[-enable_multiple_categories_per_endpoint]
[-separate_non_standard_groups]
[-separate_all_groups]
[-significant_digits digits]
[-sms_scenarios sms_scenarios_list]
[-pba_mode none | path | exhaustive | ml_exhaustive]
[-path_type summary | end]
```

Data Types

group_list	list
include_list	list
format_spec	string
file_name	string
digits	integer
sms_scenarios_list	collection

Arguments

`-delay_type max | min`

Specifies the type of delay violations to report. By default, both max (setup) and min (hold) violations are reported.

`-group group_list`

Reports violations only for the specified timing path groups. With the *-group* option, the command collects the worst path endpoint violations for each group

separately, The report therefore is generated using violations whose endpoints are unique to a path group. The same endpoint contributes to the total report for each group in which it has a violating slack. Use the *get_path_groups* command to query the path groups in the design.

`-include` *include_list*

Specifies a list of additional reporting options. The list includes one or more of the following reporting options.

- *inter_clock* - Shows the breakdown of violations for different combinations of launching and capturing clocks. By default, only the worst violating timing path to each endpoint is considered in the report. Use *per_clock_violations* to check violations specific to each clock combination.
- *non_violated* - Includes interclock combinations that have no violations, in addition to those with violations, possibly resulting in a longer report. By default, only the worst violating timing path to each endpoint is considered in the report. If a clock combination shows no violation, it means that there is no violation with a unique endpoint for that clock combination.
- *per_clock_violations* - Takes into account other paths to the same endpoint, differentiated by: timing path type (register-to-register, input-to-register, input-to-output and register-to-output), launch/capture clock combination, and path group. Using this option might lead to a longer runtime. See "Path Uniquification" below for details.
- *scenario_details* - In a distributed multi-scenario analysis (DMSA) manager process, shows the breakdown of violations by scenario in the merged report. Only the worst violating timing path to each endpoint is considered in the merged report. If a scenario sub-report shows no violation, it means that the scenario did not contribute worst violations across all scenarios currently in focus.

`-format` *format_spec*

Specifies the reporting format:

- *narrow* (default) - Generates a report with WNS-TNS-NUM values stacked in a column.
- *wide* - Generates a report with WNS-TNS-NUM values spread across a row.
- *csv* - Generates a comma-separated values (CSV) report file, which can be read into spreadsheet programs. When using this option, specify the output file name with the *-output* option. You can use the *csv* option with *narrow* or *wide* to get both an on-screen report and a CSV file.

The *narrow* and *wide* values are ignored if *-path_type end* is specified.

`-output` *`file_name`*

Specifies the file name for the generated CSV report.

`-enable_multiple_categories_per_endpoint`

Takes into account other paths to the same endpoint, differentiated by timing path type (register-to-register, input-to-register, input-to-output and register-to-output). Each path endpoint contributes the worst violation per a unique combination of timing path types. Using this option may lead to longer runtime. Using this option together with *-include per_clock_violations* does not have any additional effect.

See "Path Uniquification" below for details.

`-separate_non_standard_groups`

This formatting option divides the report into sub-reports, one for all standard path groups combined and one for each of the nonstandard path groups: **default**, **async_default**, and **clock_gating_default**. This option shows the contribution of these path group categories to the total report. If an endpoint has WNS timing path in multiple path groups, the report only takes into account the WNS of the worst path across all path groups.

`-separate_all_groups`

This formatting option divides the report into sub-reports, one for each path group, including each standard and nonstandard path group. By default, only the worst violating timing path to each endpoint is considered in the report. The values of WNS, TNS, and NUM in sub-reports add up to match the summary report for all groups. If a group sub-report shows no violation, it means that there is no violation with a unique endpoint for that path group.

The WNS path reported for each group is the overall worst negative path across all groups applied to that specific group's context. This may not be the WNS path for that specific group. To generate violation data for each group separately, use the *-group group_list* option, which guarantees reporting of the WNS of each path group.

`-significant_digits` *`digits`*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13. If you do not use this option, the number of digits is specified by the *report_default_significant_digits* variable, which is 2 by default.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only timing violations compatible to the specified SMS scenarios collection are reported. This option is available only with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Controls path-based analysis with the following modes:

- *none* (the default) - Path-based analysis is not used.
- *path* - Path-based analysis is applied over the paths that gave the worst graph-based analysis violations. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- *exhaustive* - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.
- *ml_exhaustive* - Performs machine-learning-based exhaustive path-based analysis. This mechanism deploys machine learning techniques to trade runtime versus accuracy during the design flow. Accuracy and runtime will approach *pba_mode exhaustive* as the design approaches signoff.

`-path_type summary | end`

Specifies how paths are displayed in the report. The default is *summary*, which provides a top-level summary of the timing of the design. A value of *end* provides a per-endpoint breakdown of the paths contributing to the summary report. The summary report is not displayed when a per-endpoint report is printed.

Description

The *report_global_timing* command generates a top-level summary of the timing for the design. The report shows the worst negative slack (WNS), the sum total of all negative slacks (TNS), and a number of violating endpoints (NUM).

By default, each violating endpoint is only counted one time and only one worst slack at each violating endpoint contributes to the total negative slack. With the *-group* option, each endpoint contributes to the worst negative slack and total negative slack of each group in which it has a violating slack.

The summary information is broken down into columns showing the WNS, TNS, and number of violations, both in total and subdivided into register-to-register, input-to-register, register-to-output, and input-to-output categories.

In a distributed multi-scenario analysis (DMSA) manager, the report shows a merged view of violations across all scenarios. Each unique endpoint contributes to the report only once. To see non-merged reports per scenario, use the *remote_execute -verbose {report_global_timing}* command.

If you specify the *-pba_mode exhaustive* option, path-based analysis is used to calculate the violating slack values. This option performs a path search to ensure that the returned slack values are truly the worst. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing

improvement from path-based analysis increases, the runtime for the path search also increases.

Path Uniquification

By default, the *report_global_timing* command considers the single worst path to each endpoint, determines its path type (reg-to-reg and so on), and counts it once in that category only. If there is a subcritical path to the same endpoint of a different path type, it is not counted in that category.

The *-enable_multiple_categories_per_endpoint* option causes the worst path to be counted:

- for each path type (reg-to-reg and so on)
- to each endpoint

In this case, multiple paths to an endpoint can be considered - the worst critical path of each path type is counted in that category.

The *-include {per_clock_violations}* option causes the worst path to be counted:

- for each path type (reg-to-reg and so on)
- for each launch/capture clock combination
- for each path group
- to each endpoint

This allows an endpoint to be considered in every category that it can contribute to.

Note that these options control only how paths are obtained for analysis, not how the resulting report is grouped.

Examples

The following command generates a report of violations in the current design for path groups whose name starts with "CLK".

```
pt_shell> report_global_timing -group [get_path_groups CLK*]
****************************************
Report : global_timing
        -group CLK1 CLK0
 ...
****************************************

Setup violations
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS     -20.34    -20.34      0.00     -1.19     -0.98
```

```
TNS   -6887.98  -6882.84      0.00     -3.17     -1.96
NUM        398       392         0         4         2
---------------------------------------------------------

Hold violations
---------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
---------------------------------------------------------
WNS      -3.55     -3.55      0.00      0.00      0.00
TNS    -561.17   -561.17      0.00      0.00      0.00
NUM        418       418         0         0         0
---------------------------------------------------------
```

The following command generates an interclock report showing the timing summary for each combination of launch clock and capture clock. Each sub-report starts with a "C2C" (clock-to-clock) line that identifies the launch clock -> capture clock combination. The asterisk character "*" means any clock.

```
pt_shell> report_global_timing -include {inter_clock}
****************************************
Report : global_timing
        -include {inter_clock}
 ...
****************************************

Setup violations
---------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
---------------------------------------------------------
C2C * -> *
WNS     -26.57    -26.57      0.00     -1.72     -0.98
TNS  -33377.09 -33369.93      0.00     -5.19     -1.96
NUM       1958      1950         0         6         2
---------------------------------------------------------
C2C * -> CLK0
WNS     -20.34    -20.34      0.00     -1.19     -0.98
TNS   -6889.55  -6884.41      0.00     -3.17     -1.96
NUM        403       397         0         4         2
---------------------------------------------------------
C2C CLK0 -> CLK0
WNS     -20.34    -20.34      0.00      0.00      0.00
TNS   -6884.25  -6884.25      0.00      0.00      0.00
NUM        396       396         0         0         0
---------------------------------------------------------
C2C AUDIO_SCLK_IN -> CLK0
WNS      -1.10      0.00      0.00     -1.10     -0.98
TNS      -3.06      0.00      0.00     -1.10     -1.96
NUM          3         0         0         1         2
---------------------------------------------------------
 ...
```

Feedback

The following example adds the *-include {per_clock_violations}* option. Compared to the previous example, more violations are reported because more violating paths to the same endpoint are considered.

```
pt_shell> report_global_timing -include {inter_clock
 per_clock_violations}
****************************************
Report : global_timing
        -include { inter_clock per_clock_violations }
 ...
****************************************

Setup violations
-------------------------------------------------------------
          Total    reg->reg    in->reg   reg->out    in->out
-------------------------------------------------------------
C2C * -> *
WNS       -26.57      -26.57      -2.17      -1.72      -0.98
TNS   -391171.34 -391157.25      -2.17      -9.81      -2.05
NUM         57766       57751          1         11          3
-------------------------------------------------------------
C2C * -> CLK0
WNS       -20.34      -20.34       0.00      -1.72      -0.98
TNS    -39827.79   -39817.95       0.00      -7.79      -2.05
NUM         5256        5244          0          9          3
-------------------------------------------------------------
C2C CLK0 -> CLK0
WNS       -20.34      -20.34       0.00       0.00       0.00
TNS     -6884.25    -6884.25       0.00       0.00       0.00
NUM          396         396          0          0          0
-------------------------------------------------------------
C2C ADC_PCMCLK_IN -> CLK0
WNS       -0.09        0.00       0.00       0.00      -0.09
TNS       -0.09        0.00       0.00       0.00      -0.09
NUM           1           0          0          0          1
-------------------------------------------------------------
C2C AUDIO_SCLK_IN -> CLK0
WNS       -1.10        0.00       0.00      -1.10      -0.98
TNS       -3.06        0.00       0.00      -1.10      -1.96
NUM           3           0          0          1          2
-------------------------------------------------------------
 ...
```

The following example shows the alternative reporting formats.

```
pt_shell> report_global_timing -format {wide csv} -output ./example.csv
****************************************
Report : global_timing
        -format { wide csv }
        -output example.csv
 ...
****************************************
```

```
Setup violations
          Total        |         reg->reg      |     in->reg    |
 reg->out    |      in->out    |
 WNS       TNS     NUM |  WNS       TNS    NUM  | WNS  TNS  NUM | WNS
 TNS   NUM | WNS     TNS   NUM |
-------------------------------------------------------------------------
--------------------------
-26.57 -33377.09 1958 | -26.57 -33369.93 1950 | 0.00 0.00   0 | -1.72
 -5.19   6   | -0.98 -1.96   2 |

Hold violations
          Total        |         reg->reg      |     in->reg    |
 reg->out    |      in->out    |
 WNS       TNS     NUM |  WNS       TNS    NUM  | WNS  TNS  NUM | WNS    TNS
  NUM   | WNS     TNS   NUM |
-------------------------------------------------------------------------
--------------------------
-10.89   -757.51   532 | -10.89   -757.51   532 | 0.00 0.00   0 | 0.00
 0.00   0    |  0.00   0.00    0 |

Wrote the report in CSV format into the file ./example.csv
1
pt_shell> sh cat ./example.csv
Type,Total_WNS,Total_TNS,Total_NUM,reg2reg_WNS,reg2reg_TNS,reg2reg_NUM,in
2reg_WNS,in2reg_TNS,
in2reg_NUM,reg2out_WNS,reg2out_TNS,reg2out_NUM,in2out_WNS,in2out_TNS,in2o
ut_NUM
max,-26.57,-33377.09,1958,-26.57,-33369.93,1950,0.00,0.00,0,-1.72,-5.19,6
,-0.98,-1.96,2
min,-10.89,-757.51,532,-10.89,-757.51,532,0.00,0.00,0,0.00,0.00,0,0.00,0.
00,0
```

The following example generates a report with separate results for specified path groups:

```
pt_shell> report_global_timing -group {CLK0 AUDIO_SCLK_IN} \\
            -separate_all_groups -delay_type max
 ...

Setup violations for paths in CLK0
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS     -20.34    -20.34      0.00     -1.19     -0.98
TNS   -6887.98  -6882.84      0.00     -3.17     -1.96
NUM        398       392         0         4         2
-----------------------------------------------------------

Setup violations for paths in AUDIO_SCLK_IN
-----------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-----------------------------------------------------------
WNS      -2.31     -2.31      0.00      0.00      0.00
```

```
TNS      -47.60    -47.60       0.00       0.00       0.00
NUM          32        32          0          0          0
-------------------------------------------------------------
```

The following example shows a report with results for standard path groups combined and individual nonstandard path groups separately:

```
pt_shell> report_global_timing -group {CLK0 AUDIO_SCLK_IN
 **async_default**} \\
            -separate_non_standard_groups -delay_type max
 ...

Setup violations for standard paths
-------------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS      -20.34    -20.34       0.00     -1.19     -0.98
TNS    -6935.58  -6930.45       0.00     -3.17     -1.96
NUM         430       424          0          4          2
-------------------------------------------------------------

Setup violations for paths in **async_default**
-------------------------------------------------------------
         Total  reg->reg   in->reg  reg->out   in->out
-------------------------------------------------------------
WNS      -23.71    -23.71       0.00       0.00       0.00
TNS    -1010.10  -1010.10       0.00       0.00       0.00
NUM         186       186          0          0          0
-------------------------------------------------------------
```

See Also

- report_analysis_coverage
- report_timing

---

report_gpd_config

Reports the original or modified GPD configuration options for reading parasitic data from a specified GPD directory.

Syntax

string *report_gpd_config*

```
[-type type]
[-include_starrc_options]
[-nosplit]
-gpd gpd_directory
```

Feedback

Data Types

type	string
gpd_directory	string

Arguments

`-type` *type*

Specifies the type of GPD configuration to report:

- *original* -- The original configuration of the GPD directory, as written out by the StarRC tool
- *user_defined* -- The GPD configuration specified by the *set_gpd_config* command
- *active* (default) -- The currently active GPD configuration, which is the same as the *user_defined* configuration if you have used the *set_gpd_config* command, or same as the original configuration if you have not used the *set_gpd_config* or after you have used the *reset_gpd_config* command

`-include_starrc_options`

Includes StarRC GPD configuration options in the report. This includes options not considered by the PrimeTime tool.

`-nosplit`

Prevents splitting of long lines in the report.

`-gpd` *gpd_directory*

Specifies the absolute or relative path to the GPD directory.

Description

The *report_gpd_config* command reports the configuration options for a specified Galaxy Parasitic Database (GPD) directory. The configuration options affect the reading of parasitic data from the GPD directory. You can set these options with the *set_gpd_config* command.

Examples

The following commands set and then report the current and original GPD configuration options for the GPD database called my_design1.gpd.

```
pt_shell> set_gpd_config -gpd my_design1.gpd \\
  -absolute_coupling_threshold 0.005 \\
  -relative_coupling_threshold 0.05
 1
pt_shell> report_gpd_config -gpd my_design1.gpd
 ...
```

Feedback

```
pt_shell> report_gpd_config -gpd my_design1.gpd -type original
  ...
```

See Also

- report_gpd_properties
- set_gpd_config
- reset_gpd_config

---

report_gpd_properties

Reports the layers, parasitic corners, and other information about parasitic data stored in a GPD disk directory.

Syntax

string *report_gpd_properties*

```
[-all]
[-layers]
[-parasitic_corners]
-gpd gpd_directory
[-nosplit]
```

Data Types

gpd_directory	string

Arguments

`-all`

Reports all types of information about the parasitic data stored in the specified Galaxy Parasitic Database (GPD) directory, including a summary, the layers, and the parasitic corners. By default, only the GPD summary is reported.

`-layers`

Reports only the layers of the GPD directory, in a format consistent with the layer ID number reported by the *get_ground_capacitors*, *get_coupling_capacitors*, and *get_resistors* commands.

`-parasitic_corners`

Reports only the parasitic corners of the GPD directory, whether a single corner or multiple corners.

`-gpd gpd_directory`

Specifies the absolute or relative path to the GPD directory.

Feedback

-nosplit

Prevents splitting of long lines of the report into multiple lines.

Description

This command reports the properties of parasitic data stored in a Galaxy Parasitics Database (GPD) directory. You can query the database without reading it into the PrimeTime tool.

You must specify the relative or absolute path the GPD directory. You can optionally specify whether to report a GPD summary (the default), the list of layers and their properties, the list of parasitic corners and their properties, or all of these types of information.

Examples

The following command reports a summary about the parasitic data stored in the GPD directory named mydesign.gpd in the current working directory.

```
pt_shell> report_gpd_properties -gpd mydesign.gpd
 ...
GPD Summary:
Properties                          Value
-------------------------------------------------
 ...                                 ...
number_of_nets                      288930
number_of_cells                     234730
number_of_ports                     1560
is_gpd_complete                     Yes
has_via_ladder                      No
 ...                                 ...
```

The following command reports only the layers stored in the GPD directory.

```
pt_shell> report_gpd_properties -layers -gpd mydesign.gpd
 ...
Layer information:
Name                  Properties            Value
-----------------------------------------------------------
SUBSTRATE             id                    0
SUBSTRATE             is_via                No
poly                  id                    1
poly                  is_via                No
M1                    id                    2
M1                    is_via                No
M2                    id                    3
M2                    is_via                No
 ...                   ...                   ...
```

The following command reports only the parasitic corners of the GPD directory.

Feedback

```
pt_shell> report_gpd_properties -gpd mydesign.gpd
 ...
Corner information:
Name                Properties              Value
-------------------------------------------------------------------
CMINW125            process_name            /mydata/mypara/grd.min
CMINW125            temperature             125
CMINW125            global_temperature      25
CMINB40             process_name            /mydata/mypara/grd.min
CMINB40             temperature             -40
CMINB40             global_temperature      25
CMINN25             process_name            /mydata/mypara/grd.min
CMINN25             temperature             25
CMINN25             global_temperature      25
CMAXW125            process_name            /mydata/mypara/grd.max
CMAXW125            temperature             125
CMAXW125            global_temperature      25
```

The following command reports all of the above types of data: GPD summary, parasitic corners, and layers.

```
pt_shell> report_gpd_properties -all -gpd mydesign.gpd
 ...
```

See Also

- get_coupling_capacitors
- get_gpd_corners
- get_gpd_layers
- get_ground_capacitors
- get_resistors
- report_gpd_config
- set_gpd_config
- parasitic_explorer_enable_analysis

---

report_ground_capacitors

Reports the ground capacitors for specified nets.

Syntax

status *report_ground_capacitors*

```
[-of_objects nets]
[-from node1 -to node2]
```

Feedback

Data Types

nets	list
node1	string
node2	string

Arguments

`-of_objects` *nets*

A list of one or more nets for which to report ground capacitors. At least one net is required. Glob-style wildcards (*) are supported.

This option is mutually exclusive with the *-from* and *-to* options.

`-from` *node1*

Specifies a pin, port, or net internal node. The tool reports the layer information for paths between this node and the node specified in the *-to* option. Both nodes must belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

`-to` *node2*

Specifies a pin, port, or net internal node. The tool reports the layer information for paths between this node and the node specified in the *-from* option. Both nodes must belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

Description

This command reports the ground capacitors for specified nets.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a ground capacitor report.

```
pt_shell> report_ground_capacitors -of_objects {SUM0 B0}
================================
Net: SUM0
Total capacitance: 0.013721
Report Type: Ground Capacitors
================================
Node            Layer       Capacitance       %Cc/Ct
====            =====       ===========       ======
0/33/M2/s       SUBSTRATE   0.000749          5.458786
0/33/M1/s       SUBSTRATE   0.000387          2.820494
SUM0:4          metal2      0.000247          1.800160
SUM0:5          metal2      0.002221          16.186867
================================
```

```
Net: B0
Total capacitance: 0.089779
Report Type: Ground Capacitors
================================
Node              Layer       Capacitance       %Cc/Ct
====              =====       ===========       ======
B0                metal2      0.000000          0.000000
0/38/M2/g         poly        0.000000          0.000000
0/38/M5/g         poly        0.000000          0.000000
0/54/M5/g         poly        0.000000          0.000000
B0:10             metal2      0.000082          0.091335
B0:11             metal2      0.001010          1.124985
```

---

report_gui_stroke_bindings

Print a report on the dictionaries and the stroke-command bindings they contain..

Syntax

report_gui_stroke_bindings

`[-dictionary` *`dictionary_name`*`]`

Arguments

`-dictionary` *`dictionary_name`*

Specify which dictionary should have its bindings reported. If not specified then all dictionaries will be reported.

Description

This command takes the data returned by get_gui_stroke_bindings and formats it into a report that is human-readable. The report prints out the stroke-to-command bindings for all dictionaries.

This command is defined as a part of the strokes Tcl package, and the command is automatically imported into the global namespace when the package is loaded.

Examples

```
ca_shell> report_gui_stroke_bindings

Dictionary: Graphics

stroke        cmdType       cmd
------        -------       ---
5             builtin       Pan_Center_Rect
852           builtin       Zoom_Full
258           builtin       Zoom_Full
159           builtin       Zoom_Out_Rect
357           builtin       Zoom_Out_Rect
```

```
753          builtin      Zoom_In_Rect
951          builtin      Zoom_In_Rect
654          builtin      Pan_Center_Rect
456          builtin      Pan_Center_Rect
123698741    builtin      Zoom_In_Rect
147896321    builtin      Zoom_In_Rect
s:14789      tcl_cmd      myTclCmd
14789        tcl_cmd      myTclCmd %rect
```

See Also

- set_gui_stroke_preferences
- set_gui_stroke_binding
- report_gui_stroke_builtins

---

report_gui_stroke_builtins

Print a report on the non-Tcl commands available for stroke bindings..

Syntax

report_gui_stroke_builtinss

`[-dictionary` *`dictionary_name`*`]`

Arguments

`-dictionary` *`dictionary_name`*

Specify which dictionary should have its builtins reported. If not specified then only the global builtins will be printed.

Description

This command takes the data returned by get_gui_stroke_bindings and formats it into a report that is human-readable. The report prints out the non-Tcl commands that can be used in stroke bindings.

This command is defined as a part of the strokes Tcl package, and the command is automatically imported into the global namespace when the package is loaded.

Examples

```
ca_shell> report_gui_stroke_builtins

Built-In Commands:

name
----
Zoom_In_Rect
```

Feedback

```
Zoom_Out
Pan_Center_Rect
Zoom_Out_Rect
Zoom_Full
Zoom_In
```

See Also

- set_gui_stroke_preferences
- set_gui_stroke_binding
- report_gui_stroke_bindings

---

report_hier_analysis

Reports the HyperScale configuration set by the *set_hier_config* command, or port abstractions set by the *set_port_abstraction* command.

Syntax

status *report_hier_analysis*

```
[-config]
[-port_abstraction]
[-parasitics]
[-nosplit]
[-verbose]
```

Arguments

`-config`

Reports the HyperScale configuration set by the *set_hier_config* command, including the HyperScale cell names, timestampw, and directory paths. This is the default report type when the *report_hier_analysis* command is used without options.

`-port_abstraction`

Reports any port abstraction overrides that have been applied to the ports of the HyperScale blocks by the *set_port_abstraction* command. "Port abstraction" refers to the removal of part or all of the logic fanin or fanout of a port.

`-parasitics`

Reports information about the parasitics files associated with the different levels of hierarchical analysis.

`-nosplit`

Prevents splitting of long lines of text in the report.

`-verbose`

Reports more detailed messages.

Description

The *report_hier_analysis* command reports HyperScale configuration information.

There are three reporting options available:

- *-config*
- *-port_abstraction*
- *-parasitics*

You can specify any combination of these options to report the desired information. If no option is specified, a configuration report (*-config*) is given.

The following sections describe the reports generated by these options.

HyperScale Block Configuration

A block configuration report shows the cell names, creation timestamps, and directory paths of the HyperScale blocks in the design, as defined by the *set_hier_config* command.

Port Abstraction

A port abstraction report shows the block ports that have been abstracted and the reason for each abstraction. When a port is abstracted in a HyperScale block, some portion of the port's fanout or fanin logic cone is made invisible at higher levels of hierarchy.

For example, each clock port is abstracted by default. In this case, the logic in the fanout cone that leads only to internal registers of the block is not visible at higher levels of hierarchical analysis. Internal registers are registers that cannot be reached from a non-abstracted port through combinational logic.

If you need timing data for an object that is not visible due to block-level port abstraction, you can either access that data during block-level analysis, or you can use the *set_port_abstraction* command to override the default behavior and prevent abstraction of the port for higher-level analysis.

Parasitics Check

A parasitic check report shows information about the parasitics files associated with the different levels of hierarchical analysis, such as the file names, corner names, operating temperatures, program names and versions, and parasitics tech files.

Feedback

Examples

The following command reports the HyperScale block configuration:

```
pt_shell> report_hier_analysis -config
 ...
 Cell            Name     Timestamp           HyperScale path
 ------------------------------------------------------------------
 blkA            default  2017-03-02 10:42:31 /disk1/blkA/HS_DATA/
 <design>        default  2017-03-02 10:44:38 /disk1/TOP/HS_DATA/
```

The following command reports the HyperScale port abstractions:

```
pt_shell> report_hier_analysis -port_abstraction
 ...
 Abstraction codes:
   clk  Clock port
   lfi  Large fanout input port

 Port            Abstraction code
 ------------------------------------------------
 pclk            clk
 scan_en         lfi
 sdr_clk         clk
 sys_clk         clk
```

The following command reports all parasitics file information:

```
pt_shell> report_hier_analysis -parasitics
 ...
 SPEF file  Corner name  Operating temp  Program name   Program version
 Parasitics Tech file
 ------------------------------------------------------------------------
---------------------
 top.spef   RCworstt     -90             TOP_TOOL_NAME  2000.10-DEV
  /grd/file/filet.nxtrg
 blk.spef   RCworstb     -80             BLK_TOOL_NAME  P-2019.03-SP3
  /grd/file/fileb.nxtrg
 ...
```

See Also

- set_hier_config
- set_port_abstraction
- set_dont_override

---

report_hierarchy

Reports the reference hierarchy of the current instance or current design.

Feedback

Syntax

string *report_hierarchy*

```
[-full]
[-noleaf]
[-nosplit]
```

Arguments

`-full`

Displays the full hierarchy. By default, components of submodules in multiple locations in an hierarchy are listed only once. An ellipsis (...) indicates the contents of a previously displayed module.

`-noleaf`

Does not display the leaf library cells.

`-nosplit`

Does not split lines if column overflows.

Description

Displays the indented reference hierarchy of the current instance or the current design. If the current instance is set, the report is generated relative to that instance; otherwise, the report is generated for the current design.

Examples

The following command reports the full hierarchy of the design.

```
pt_shell> report_hierarchy -full

****************************************
Report : hierarchy
Design : middle
****************************************

     FD2                              tech_lib
     ND2                              tech_lib
     inter
          low
              NR4                     tech_lib
     low
          NR4                         tech_lib
```

See Also

- report_reference
- report_cell

Feedback

report_host_usage

Creates a detailed report that describes all host options that you have created.

Syntax

status *report_host_usage*

```
[-verbose]
[-nosplit]
[host_option_names]
```

Data Types

host_option_names	list

Arguments

`-verbose`

Generates a more detailed report.

`-nosplit`

Does not split lines if columns overflow.

host_option_names

Reports the specified list of host options, which were previously defined by the *set_hosts_options* command. If you do not specify a list of host options, the command reports all host options by default.

Description

The *report_host_usage* command reports information about the specified sets of host options. The command also reports peak memory and CPU usage for the local process and all online distributed processes.

Running the *report_host_usage* command before starting any hosts reports the following basic information:

- Options Name - The name of the host options set.
- Host Name - The host on which the processes are to be launched. If the processes are to be launched on a managed compute resource, the string "farm" is shown.
- Num Processes - The number of processes to be launched.
- Protocol - The type of the communication protocol specified by the *-protocol* option of the *set_host_options* command. The default value of the protocol is *auto*.

The *-verbose* option reports the following additional information:

- Action and Command - The actions and associated commands that you can perform for the host options. SUBMIT indicates the command to submit a process for execution. TERMINATE indicates the command to terminate a job that has not started.
- Protocol - The type of the communication protocol that is actually used to communicate between the manager process and remote hosts.
- # - The internal instance number of the process. When each worker instance is launched by the *start_hosts* command it is assigned an instance number. This is to allow you to relate the process to the output of the *start_hosts* command.
- Host Name - The name of the actual host on which the process is running.
- Job ID - If the options specified access a managed resource, such as LSF or SGE, this is the ID of the job submitted to the farm. If this value is -1, the job ID either cannot be determined or the process was shutdown in which case the ID is no longer valid. For non-managed resources, a symbol "-" is shown.
- Process ID - The ID of the worker PrimeTime process. If this value is -1, the process ID either could not be determined or the process was shutdown in which case the ID is no longer valid.
- Status - The status of the worker process. The possible values are

```
LAUNCHED      : The process is launched but not ready for use yet.
ONLINE        : The process is online and ready for use.
SHUTDOWN      : The process was shut down.
FATALed       : The process abnormally terminated.
MISSALIGNED   : The manager and worker processes are different
 versions
                of PrimeTime and the worker process is not usable.
SHUTTING_DOWN : The process is shutting down.
```

Usage Limits (Cores)

This section of the report shows details about the cores usage limits applied to each set of host options. The first entry in this table describes the limits on the current local process. The cores limits columns are

- Effective

  This is the effective cores usage limit to be honored by PrimeTime. This value is computed given the user and hardware limits.

If you use the *-verbose* option, the following columns are also shown:

- User-set - The *max_cores* that you set in the corresponding *set_host_options* command.
- Hardware - The number of logical cores on the target hardware.

Memory Usage

This section of the report shows details about the peak memory usage of all online processes. The first entry in this table describes the current local process. The memory usage columns are

- Memory (MB) - The peak memory (in megabytes) usage of the local and all distributed pt_shell processes currently online.

Performance

This section of the report shows details the CPU and Elapsed time measurements (in seconds) of all online processes. The first entry in this table describes the current local process. The performance columns are

- CPU Time (s) - This shows the total CPU time usage for each online process.
- Elapsed Time (s) - This shows the total elapsed time for each online process.

Examples

In the following example, PrimeTime is launched onto a 16-core machine. Initially, reporting usage shows that the user-specified local process is effectively limited to 4 cores even though no user limit is set. The subsequent *set_host_options* command increases the number of cores to 8.

```
pt_shell> report_host_usage -verbose
****************************************
Report : host_usage
         -verbose
 ...
****************************************


  Usage limits (cores)

  Options Name     #                 User-set  Hardware      Effective
  ------------------------------------------------------------------------
  (local process)  -                 4         16            4
  ------------------------------------------------------------------------
  Total                                                      4

  Memory usage

  Options Name     #                           Memory (MB)
```

```
  ------------------------------------------------------------------
  (local process)  -                            287.65

  Performance

  Options Name     #    CPU Time (s)           Elapsed Time (s)
  ------------------------------------------------------------------
  (local process)  -    2                      6

1

pt_shell> set_host_options -max_cores 8
Information: The max_cores limit for the local (current) process has been
  modified by 4 to 8. (CMCR-103)
1
pt_shell> report_host_usage -verbose
****************************************
Report : host_usage
         -verbose
****************************************


  Usage limits (cores)

  Options Name     #                 User-set  Hardware      Effective
  ------------------------------------------------------------------
  (local process)  -                 8         16            8
  ------------------------------------------------------------------
  Total                                                      8

  Memory usage

  Options Name     #                            Memory (MB)
  ------------------------------------------------------------------
  (local process)  -                            364.57

  Performance

  Options Name     #    CPU Time (s)           Elapsed Time (s)
  ------------------------------------------------------------------
  (local process)  -    2                      85

1
```

For more examples, see the man page for the *set_host_options* command.

See Also

- remove_host_options
- set_host_options

Feedback

- start_hosts
- stop_hosts

report_ideal_network

Displays information about ports, pins, nets, and cells on ideal networks in the current design.

Syntax

status *report_ideal_network*

```
[-net]
[-cell]
[-load_pin]
[-timing]
[object_list]
```

Data Types

object_list list

Arguments

`-net`

Displays all nets in the specified ideal networks. By default, nets are displayed for all ideal networks.

`-cell`

Displays all cells in the specified ideal networks. By default, cells are displayed for all ideal networks.

`-load_pin`

Specifies to display load pins (boundary pins) of the specified ideal networks along with any ideal timing set on them using the *set_ideal_latency* and *set_ideal_transition* commands. By default, load pins are displayed for all ideal networks.

`-timing`

Specifies to display all internal pins (non-source and non-boundary pins) in the specified ideal networks that have ideal timing set on them using the *set_ideal_latency* and *set_ideal_transition* commands. By default, internal pins with timing are displayed for all ideal networks.

object_list

Defines a list of source ports or source pins of the specified ideal networks to be displayed. By default, the source pins and source ports for all ideal networks in the current design are displayed.

Description

Displays information about the ideal networks in the current design. If no arguments are specified, all ideal network sources in the current design are displayed. If you specify a list of source pins or ports, the command displays information about the ideal networks emanating from these objects.

The "Latency" and "Transition" columns show the minimum and maximum values set for both rising and falling edges. Set these values with the *set_ideal_latency* and *set_ideal_transition* commands.

Examples

The following example sets up an ideal network, applies ideal latency and ideal transition values and shows the output of the ideal network report.

```
pt_shell> set_ideal_network {p_in in1}
pt_shell> set_ideal_latency    -min  1.12 in1
pt_shell> set_ideal_transition -max  9.87 in1
pt_shell> set_ideal_latency    -min  1.34 n0_i/Z
pt_shell> set_ideal_transition -rise 5.62 n3_i/B
pt_shell> report_ideal_network -timing -load_pin -cell -net

****************************************
Report : ideal_network
        -timing
        -load_pin
        -net
        -cell
 ...
****************************************

Source ports                    Latency                        Transition
and pins                  Rise            Fall            Rise
 Fall
                        min    max      min    max      min    max
 min    max
-------------------------------------------------------------------------
------------
middle/in1             1.12    --      1.12    --       --    9.87
 --    9.87
middle/p_in             --     --       --     --       --     --
 --     --

Internal pins                   Latency                        Transition
```

```
with ideal timing          Rise             Fall             Rise
 Fall
                         min    max       min    max       min    max
 min    max
-------------------------------------------------------------------------
------------
n3_i/B                    --     --        --     --       5.62   5.62
 --     --
n0_i/Z                   1.34    --       1.34    --        --     --
 --     --

Boundary pins                      Latency                        Transition
                            Rise             Fall             Rise
 Fall
                         min    max       min    max       min    max
 min    max
-------------------------------------------------------------------------
------------
o_reg3/D                  --     --        --     --        --     --
 --     --
middle/p_out              --     --        --     --        --     --
 --     --

Nets
-------------------------------------------------------------------------
------------
in1
p_in
p_out
n2
n1
n0

Cells
-------------------------------------------------------------------------
------------
n3_i
n2_i
n1_i
n0_i
```

The following example shows the output of the ideal network report on specified objects.

```
pt_shell> report_ideal_network -timing -load_pin -cell -net {in1}

****************************************
Report : ideal_network
        -timing
        -load_pin
        -net
        -cell
 ...
****************************************
```

```
Source ports                        Latency                          Transition
and pins                    Rise             Fall             Rise
 Fall
                          min     max      min     max      min     max
 min     max
-------------------------------------------------------------------------------
------------
middle/in1                1.12     --      1.12     --       --     9.87
 --     9.87

Boundary pins                       Latency                          Transition
                            Rise             Fall             Rise
 Fall
                          min     max      min     max      min     max
 min     max
-------------------------------------------------------------------------------
------------
o_reg3/D                   --      --       --      --       --      --
 --      --

Nets
-------------------------------------------------------------------------------
------------
in1
```

See Also

- remove_ideal_network
- report_constraint
- set_ideal_latency
- set_ideal_network
- set_ideal_transition

---

report_implement_options

Reports options specified by the *set_implement_options* command.

Syntax

string *report_implement_options*

Arguments

None

Feedback

Description

The *report_implement_options* command reports all PrimeECO options specified by the *set_implement_options* command to configure physical signoff ECO flow using IC Compiler II and StarRC.

Examples

The following example reports the ECO options previously specified by the *set_implement_options* command.

```
prompt> report_implement_options
Implementation Options
------------------------------------------------------------
ICC2 executable          : /path/to/icc2_shell
StarRC executable        : /path/to/StarXtract
StarRC mode              : full
Working directory        : ./dcs_work_dir
Physical ICC2 library    : ./design.nlib
Physical ICC2 block      : pre_eco_block
Current working block    : ecoBlock_pid6163

1
```

See Also

- check_eco
- check_routes
- implement_eco
- reset_implement_options
- save_block
- set_eco_options
- set_implement_options
- write_implement_changes

---

report_instances

Creates a collection of the instances (cells) associated with one or more nets.

Syntax

status *report_instances*

```
[-filter expression]
```

Feedback

Data Types

expression	list

Arguments

`-filter` *expression*

A list of one or more nets for which to report cells. At least one net is required. Glob-style wildcards (*) are supported.

Description

This command creates a collection of the instances (cells) associated with one or more nets.

The command reports all attributes of the instances with the same format as the instance section of a SPEF file. It reports instances with their name, pin or port names, model name, and their properties. It also provides the location information of a device at the end of the report, if the device location is available.

This command is supported only for transistor-level GPDs.

Examples

The following examples shows the *report_instances* command report:

```
pt_shell> report_instances -filter {name==0\\/33\\/M1} _se14
****************************************
Report : Instances summary
Design : add4
Version: R-2020.09
Date   : Tue Aug 18 15:11:19 2020
****************************************
Instance lines
==============================

M0/33/M1 GND 0/33/M1:g 0/33/M1:s GND n w=13.000u 1=1.000u AD=39p AS=39p
 PD=32u PS=32u
```

report_ivm

Reports via variation tables previously read in by the *read_ivm* command.

Syntax

```
status report_ivm
   [-layer_name string]
   [-summary]
```

Feedback

Data Types

string string

Arguments

-layer_name

Reports variation tables only for the specified layer. By default, the command reports all layers.

-summary

Reports a summary of the all the variation tables that have been read in by the *read_ivm* command.

Description

The *report_ivm* command displays the user-specified via variation information previously read in by the *read_ivm* command. The report format is similar that of the data file.

Examples

The following command reports the via variation data for only the V2 layer.

```
pt_shell> report_ivm -layer_name V2
 ...
Number of VIA side-file tables: 3

Layer Name: V2
area            : 0.000400 0.000800 0.001600
table_min_sigma : 0.190000 0.159000 0.128000
table_max_sigma : 0.224000 0.208000 0.192000
table_meanshift : 0.000000 0.000000 0.000000
table_stdev     : 0.000000 0.000000 0.000000
```

See Also

- read_ivm
- remove_ivm
- timing_enable_via_variation

---

report_latch_loop_groups

Reports on latch data pins involved in loops of transparent latches.

Syntax

string *report_latch_loop_groups*

```
[-of_objects pin_list]
[-loop_breakers_only]
[-path_breakers_only]
[-nosplit]
```

Data Types

pin_list list

Arguments

`-of_objects` *pin_list*

Specifies a collection of data pins of transparent latches. Only pins of loop groups containing the specified pins are reported. By default, the command reports on all latch data pins involved with transparent latch loops, as well as other latch data pins outside loops with the *is_latch_loop_breaker* pin attribute set to *true*.

`-loop_breakers_only`

Specifies that the report should contain only pins that are loop breaker latch data pins and should be pins truly contained in transparent latch loops. By default, all transparent latch data pins within the loop group are reported, along with other latch data pins with the *is_latch_loop_breaker* pin attribute set to *true*.

`-path_breakers_only`

Specifies that the report should contain only pins with the *is_latch_loop_breaker* pin attribute set to *true*. By default, all transparent latch data pins within the loop group are returned in the collections.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

Description

When sequential loops of transparent latches exist in a design, a loop group containing all the latch data pins of intersecting loops is formed. The *report_latch_loop_groups* command analyzes the design and reports on latch data pins involved with loops. The report also includes some latch data pins outside loops. Combinational pins are not included in the results.

Latch data pins outside loops can be reported if the latch is being treated by the tool as a latch loop breaker data pin, even if it is not inside a latch loop. This can happen when either the user requests the pin to be treated this way using the *set_latch_loop_breaker* command, or the tool has selected the pin to be treated as a loop breaker data pin to

save runtime. For more information about how this can happen, see the man page of the *timing_through_path_max_segments* variable.

The report lists the name of the latch data pin, a number identifying which latch loop group the pin is part of, and attributes describing if the pin is a loop breaker latch, and why.

The number identifying which latch loop group the pin is part of is an arbitrary number, except that within a report, the latch data pins that are part of the same loop group have the same number. For latch data pins that are not part of a loop group, "NA" is shown instead of a loop group number.

Before using the *report_latch_loop_groups* command, you must set the *timing_enable_through_paths* variable to *true*.

Examples

The following example uses the *report_latch_loop_groups* command to report latch loops in the design:

```
pt_shell> report_latch_loop_groups
****************************************
Report : latch loop groups
 ...
****************************************
Attributes
   b loop breaker d pin
   p long path breaker d pin, but not in a loop
   u user requested to be a loop breaker using set_latch_loop_breaker
   a user requested to avoid with set_latch_loop_breaker

  Latch                                 Latch Loop    Attributes
  D pin                                 Group
-------------------------------------------------------------------
DUT/Latch1/D                            NA            pu
DUT/Latch2/D                            1             b
DUT/Latch3/D                            1
DUT/Latch4/D                            2             bu
DUT/Latch5/D                            2             a
```

See Also

- report_latch_loop_groups
- timing_enable_through_paths
- timing_through_path_max_segments

---

report_length_layerwise

Reports the distribution of length with respect to layers for specified nets.

Feedback

Syntax

status *report_length_layerwise*

`-of_objects nets`

Data Types

nets	list

Arguments

`-of_objects nets`

A list of one or more nets for which to report lengths. At least one net is required. Glob-style wildcards (*) are supported.

Description

This command reports the distribution of length with respect to layers for specified nets.

It requires that the original extraction be performed with the *NETLIST_TAIL_COMMENTS: YES* command to save the required information.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a net length report. The report contains a section for each specified net with a list of layers and the length of the specified net on each layer.

```
pt_shell> report_length_layerwise -of_objects {SUM0 B0}
================================
Net: SUM0
Report: Layerwise length of net
================================
metal1 21u
metal2 55.5u
================================
Net: B0
Report: Layerwise length of net
================================
metal1 252.5u
metal2 113u
poly   189u
```

---

report_lib

Reports library information.

Syntax

string *report_lib*

Feedback

```
[-nosplit]
[-timing_arcs]
[-power_arcs]
library
[lib_cell_list]
```

Data Types

```
library              list
lib_cell_list             list
```

Arguments

`-nosplit`

Prevents splitting lines if a column overflows. Some of the information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

`-timing_arcs`

Displays timing arc information. Indicates to list all cell timing arcs.

`-power_arcs`

Displays power arc information. Indicates to list all cell power arcs.

`library`

A pattern matching a single library or a collection containing one library. The library must have been read by using the *read_db* command.

`lib_cell_list`

Displays a list of library cells about which information is reported. The default is to report information about all cells in the technology library. Each element in this list is either a collection of library cells or a pattern that matches library cells in the *library* option.

Description

A library report displays library information including a list of operating conditions, wire load models, and available library cells. The *-timing_arcs* option lists detailed timing arc information for each library cell, while the *-power_arcs* option lists detailed power arc information. Note: Currently, the *-power_arcs* and *-timing_arcs* options are mutually exclusive. The *report_lib* command also indicates if the library is a maximum library in a max/min relationship to a minimum library created by the *set_min_library* command.

To generate a report, the specified library must be loaded into pt_shell.

Feedback

Examples

This example shows the default library report.

```
pt_shell> read_db tech_lib.db

pt_shell> report_lib tech_lib

****************************************
Report : library
Library: tech_lib
****************************************

Time Unit                    : 1 ns
Capacitance Unit             : 1 pF
Default Wire Load Mode       : top
Default Wire Load Model      : Not specified.
Default Wire Selection Group: Not specified.

Operating Conditions:

   Name                 Process     Temp        Voltage     Tree Type

 ------------------------------------------------------------------------
---
   BCCOM                  0.60        0.00        5.25     best_case
   TYPICAL                1.00       25.00        5.00     balanced_case
   WCCOM                  1.50       70.00        4.75     worst_case


Wire Load Models:

   Name

 ------------------------------------------------------------------------
---
   small_wl
   medium_wl
   large_wl

Library Cells:

   Attributes:
    b - black box (function unknown)

   Lib Cell  Attributes
   ------------------------------
   AN2
   BUFA
   BUFB
   FF
   IV
   LTCH
```

```
   MUX21
   OR2
```

In this example, timing arc information is shown for a list of library cells.

```
pt_shell> report_lib -timing_arcs tech_lib {AN2 OR2}

****************************************
Report : library
         -timing_arcs
Library: tech_lib
****************************************

Library Cells:

   Attributes:
    b - black box (function unknown)

                              Arc                    Arc Pins
   Lib Cell  Attributes     #  Sense/Type       From         To          When

 ---------------------------------------------------------------------------
----
   AN2                      0  unate            A            Z
                            1  unate            B            Z

   OR2                      0  unate            A            Z
                            1  unate            B            Z
```

In the example below, power arc information is shown for a list of library cells.

```
pt_shell> report_lib -power_arcs tech_lib {AN2 OR2}

****************************************
Report : library
         -power_arcs
Library: tech_lib
****************************************

Time Unit                                  : 1 ns
Capacitance Unit                           : 1 pF
Pulling Resistance Unit                    : 10ohm
Voltage Unit                               : 1 V
Current Unit                               : 1e-06 A
Leakage Power Unit                         : 1e-06 W

Scaling Factors:

    *Global*

 ---------------------------------------------------------------------------
    k_process_rise_propagation             :           0.000000
    k_volt_rise_propagation                :           0.000000
```

```
    k_temp_rise_propagation                 :           0.000000
    k_process_fall_propagation              :           0.000000
    k_volt_fall_propagation                 :           0.000000
    k_temp_fall_propagation                 :           0.000000
    k_process_rise_transition               :           0.000000
    k_volt_rise_transition                  :           0.000000
    k_temp_rise_transition                  :           0.000000
    k_process_fall_transition               :           0.000000
    k_volt_fall_transition                  :           0.000000
    k_temp_fall_transition                  :           0.000000
    k_process_cell_rise                     :           0.000000
    k_volt_cell_rise                        :           0.000000
    k_temp_cell_rise                        :           0.000000
    k_process_cell_fall                     :           0.000000
    k_volt_cell_fall                        :           0.000000
    k_temp_cell_fall                        :           0.000000
    k_process_internal_power                :           0.000000
    k_volt_internal_power                   :           0.000000
    k_temp_internal_power                   :           0.000000
    k_process_cell_leakage_power            :           0.000000
    k_volt_cell_leakage_power               :           0.000000
    k_temp_cell_leakage_power               :           0.000000

    AN2_factors

-------------------------------------------------------------------------
    k_process_rise_propagation              :           0.000000
    k_volt_rise_propagation                 :           0.000000
    k_temp_rise_propagation                 :           0.000000
    k_process_fall_propagation              :           0.000000
    k_volt_fall_propagation                 :           0.000000
    k_temp_fall_propagation                 :           0.000000
    k_process_rise_transition               :           0.000000
    k_volt_rise_transition                  :           0.000000
    k_temp_rise_transition                  :           0.000000
    k_process_fall_transition               :           0.000000
    k_volt_fall_transition                  :           0.000000
    k_temp_fall_transition                  :           0.000000
    k_process_cell_rise                     :           1.000000
    k_volt_cell_rise                        :          -0.440000
    k_temp_cell_rise                        :           0.002120
    k_process_cell_fall                     :           1.000000
    k_volt_cell_fall                        :          -0.440000
    k_temp_cell_fall                        :           0.002120
    k_process_internal_power                :           0.000000
    k_volt_internal_power                   :           0.000000
    k_temp_internal_power                   :           0.000000
    k_process_cell_leakage_power            :           0.000000
    k_volt_cell_leakage_power               :           0.000000
    k_temp_cell_leakage_power               :           0.000000

    OR2_factors
```

```
 ------------------------------------------------------------------------
    k_process_rise_propagation                :            0.000000
    k_volt_rise_propagation                   :            0.000000
    k_temp_rise_propagation                   :            0.000000
    k_process_fall_propagation                :            0.000000
    k_volt_fall_propagation                   :            0.000000
    k_temp_fall_propagation                   :            0.000000
    k_process_rise_transition                 :            0.000000
    k_volt_rise_transition                    :            0.000000
    k_temp_rise_transition                    :            0.000000
    k_process_fall_transition                 :            0.000000
    k_volt_fall_transition                    :            0.000000
    k_temp_fall_transition                    :            0.000000
    k_process_cell_rise                       :            1.000000
    k_volt_cell_rise                          :           -0.440000
    k_temp_cell_rise                          :            0.002120
    k_process_cell_fall                       :            1.000000
    k_volt_cell_fall                          :           -0.440000
    k_temp_cell_fall                          :            0.002120
    k_process_internal_power                  :            0.000000
    k_volt_internal_power                     :            0.000000
    k_temp_internal_power                     :            0.000000
    k_process_cell_leakage_power              :            0.000000
    k_volt_cell_leakage_power                 :            0.000000
    k_temp_cell_leakage_power                 :            0.000000

Operating Conditions:

    Name                Process     Temp        Voltage     Tree Type

 ------------------------------------------------------------------------
    <lib_default>          1.00       27.00        2.50     balanced_case
    B                      0.63      -30.00        2.80     balanced_case
    BB                     0.63      -30.00        2.80     best_case
    BG                     0.63      -30.00        2.75     balanced_case
    TYPICAL                1.00       27.00        2.50     balanced_case
    W                      1.37      125.00        2.20     balanced_case
    WGSM                   1.37       85.00        2.25     balanced_case
    WW                     1.37      125.00        2.20     worst_case

Input Voltages:

    No input_voltage groups specified.

Output Voltages:

    No output_voltage groups specified.


Wire Loading Model:

    No wire loading specified.
```

```
Wire Loading Model Mode: top.

Delay Threshold Trip-Points
    input_threshold_pct_rise                : NOT SPECIFIED
    output_threshold_pct_rise               : NOT SPECIFIED
    input_threshold_pct_fall                : NOT SPECIFIED
    output_threshold_pct_fall               : NOT SPECIFIED


Slew Threshold Trip-Points
    slew_lower_threshold_pct_rise           : NOT SPECIFIED
    slew_upper_threshold_pct_rise           : NOT SPECIFIED
    slew_lower_threshold_pct_fall           : NOT SPECIFIED
    slew_upper_threshold_pct_fall           : NOT SPECIFIED

    slew_derate_from_lib                    : NOT SPECIFIED


Power Supply Group:

    No power supply group specified.

default_cell_leakage_power                  : 0
default_leakage_power_density               : NOT SPECIFIED

Power Information:

    Attributes:
       a - average power specification
       i - internal power
       l - leakage power
      rf - rise and fall power specification

                                        Power          Source
    Cell         Attributes  #          Toggling pin   of path  When

 ------------------------------------------------------------------------
----
    AN2          l           0
                 i,rf        1          A                       !B
                 i,rf        2          B                       !A
                 i,rf        3          Z              A        B
                 i,rf        4          Z              B        A
                 i,a         5          Z
    OR2          l           0
                 i,rf        1          B                       A
                 i,rf        2          A                       B
                 i,rf        3          Z              B        !A
                 i,rf        4          Z              A        !B
                 i,a         5          Z
```

Feedback

See Also

- read_db
- set_min_library

---

report_lib_groups

Generates a report of library groups.

Syntax

int *report_lib_groups*

```
-scaling
[-objects]
[-show]
[-nosplit]
[object_list]
[show_list]
```

Data Types

```
object_list       list
show_list       list
```

Arguments

`-scaling`

Reports the list of library groups defined for scaling. Scaling library groups can be defined using the *define_scaling_lib_group* command.

`-objects`

Reports the list of library groups in the *object_list* option. Specify this option if you want to report the library groups of some specific cell, library cell or library.

`-show`

Reports the list of options in the *show_list* option. Specify this option if you want detailed information for the libraries in the scaling group, such as voltage and temperature. If this option is used, you must also specify the *show_list* option.

`-nosplit`

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`object_list`

Specifies a list of cell, library cell or library. Reports the specific library groups that the specified object list belong.You can use any combination of these type of objects. The *object_list* option cannot be empty. If the *-objects* option is specified, you must also specify a non-empty *object_list*.

`show_list`

Reports the specific options that need to be reported. The *show_list* option can only contain options from the list {temperature, voltage, extended_name}. You can use any combination of these options. The *show_list* option cannot be empty. If the *-show* option is specified, you must also specify a non-empty *show_list*.

Description

Generates a report of the scaling library groups specified in the design.

The *scaling* option allows you to view all the scaling library groups specified using the *define_scaling_lib_group* command.

You can obtain detailed information about the libraries by using the *-show* option. The *-scaling* option is required.

Examples

The following example shows the scaling relationships that exist in the design.

```
pt_shell> report_lib_groups -scaling
****************************************
Report : lib_groups
         -scaling
Design : test
Version: Y-2006.06-Alpha-DEV
Date   : Wed Mar  8 11:22:51 2006
****************************************

Group    Library
------------------------------------------------------------
Group 1
         lib_0.95
         lib_0.85

1
```

The following example shows how you can use the *-show* option to obtain more detailed library information.

```
pt_shell> report_lib_groups -scaling -show {voltage temperature}
****************************************
Report : lib_groups
         -scaling
```

Feedback

```
          -show
Design : test
Version: Y-2006.06
Date   : Wed Mar  8 11:50:28 2006
****************************************

Group    Library      Temperature   Voltage
--------------------------------------------------------------------
Group 1
          lib_0.95     125.00        0.95
          lib_0.85     125.00        0.85

1
```

The following example shows how you can use the *-objects* option to obtain more detailed library information.

```
pt_shell> report_lib_groups -scaling -objects {tc300c_1.05} -show
 {process}
****************************************
Report : lib_groups
 -scaling
 -show
 -objects
Design : test
Version: V-2023.09-DEV-20230111
Date   : Tue Jan 24 13:41:50 2023
****************************************

Group    Library        Process
--------------------------------------------------------------------------------
------
Group 2
          tc300c_1.05       1.00
          tc300c_0.85       1.00
1
```

See Also

- define_scaling_lib_group
- report_lib

---

report_license_limit

Reports the currently defined license limits.

Syntax

```
status report_license_limit
   feature_list
```

Feedback

Data Types

feature_list list

Arguments

feature_list

Shows the limits set on the specified features. If you do not use this option, the command lists all features for which limits are set.

Description

The *report_license_limit* command shows the limits on the number of licenses that were previously set by the *set_license_limit* command.

Examples

In the following example, the upper limit on the number of PrimeTime licenses was set to seven and the number of PrimeTime SI licenses was set to four:

```
pt_shell> report_license_limit
****************************************
Report : report_license_limit
 ...
****************************************

License limits set on these features:

Feature name                                Limit
------------------------------------------------------------
PrimeTime-SI                                4
PrimeTime                                   7

1
```

See Also

- get_license
- list_licenses
- remove_license
- remove_license_limit
- report_multi_scenario_design
- set_license_limit

Feedback

report_memory_input_gate_savings

RTL memory input gate savings.

Syntax

string *report_memory_input_gate_savings*

```
[-nosplit]
[-debug_level debug_level]
[cell_list]
```

Data Types

```
debug_level     integer
cell_list       list
```

Arguments

`-nosplit`

Use this option to prevent line-splitting or section-breaking. By default, if the information for a given field exceeds its fixed column width, the next field begins on a new line, starting in the correct column (line-splitting). By default, if the information for one line exceeds the 80 character limit, the report is broken into two sections, each containing part of the information (section-breaking).

`-debug_level` *debug_level*

Use this option to report verbose information about the memory cell and its pins.

`cell_list`

This command expects user to provide memory instance name for analysis. If no instance is specified all memory instances will be analysed.

Description

This command detects memory inputs without data gating logic at RTL level.

The command estimates the power savings when the memory inputs are gated and report such memory instances along with RTL line information.

Power savings will be obtained by reduced activity on the input pins using gating.

The command only works in the average power analysis mode.

The command reports current memory instance power, estimated power using new enable signal and power savings.

Feedback

Examples

The following example shows the output report of *report_memory_input_gate_savings* command.

```
pwr_shell> report_memory_input_gate_savings
****************************************
Report : report_memory_input_gate_savings
Design : wrapper_tcbram
Version: U-2022.12-SP3
Date   : Tue May 23 13:06:15
****************************************

Memory                          Memory                          Current
  Estimated      Power            Gating          Current        Estimated
    Current        Estimated
Instance                        Cell                            Power
  Power          Savings          Enable          Read Cond.     Read Cond.
    Write Cond.    Write Cond.
Name                            Name                            (W)
  (W)            (W)              Probability     TR             TR
    TR             TR
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
ram_inst/i_sram_tcbram          sram32x64_hscm4                 0.01366
  0.01149        0.00217          0.5             0.15           0.1595
    0.15           0.08442
ram_inst1/i_sram_tcbram         sram32x64_hscm4                 0.014
  0.01183        0.00217          0.5             0.15           0.1595
    0.15           0.08442
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
1
```

See Also

- set_memory_cell_info
- compute_metrics
- update_metrics

---

report_memory_redundant_read_cycles

Analyzes input RTL FSDB for memory cells and reports redundant read cycle power.

Syntax

string *report_memory_redundant_read_cycles*

```
[object_list]
```

Data Types

object_list list

Arguments

object_list

This command expects user to provide memory instance name for analysis. If no instance is specified all memory instances will be analysed.

Description

This command analyzes input RTL FSDB or VCD waveform and reports the redundant cycles of Read Enable signal. The command reads the input RTL FSDB and uses cycle power analysis to estimate the time windows in which read address is constant and read data has been valid for clock cycle. So, *Read Enable* signal can be disabled for the subsequent clock cycles, thereby reducing memory instance power.

The command only works in the average power analysis mode.

The command dumps out an output FSDB file which shows original *Read Enable* signal as well as estimated *Read Enable* signal after redundancy analysis. Output FSDB also contains original input and output signals of the memory instances so user can validate and change RTL code.

The command reports current, modified and improvement in Read Access Rate of *Read Enable* signal. Read Access Rate is defined as Static Probability of *Read Enable* input as percentage. The command also dumps modified *Read Enable* Toggle Rate after redundancy analysis. This can be used to validate estimated power of the memory instance using estimated enable signal.

The command reports current memory instance power, estimated power using new enable signal and power savings.

User can use output FSDB and command output report to modify user's RTL and input FSDB for making architecture-level decisions and improving memory power.

Examples

The following example shows the output report of *report_memory_redundant_read_cycles* command.

```
pwr_shell> report_memory_redundant_read_cycles ram_inst1/i_sram_tcbram
****************************************
Report : report_memory_redundant_read_cycles
Design : wrapper_tcbram
Version: U-2022.12-SP3
Date   : Tue May 23 13:19:30
****************************************
```

```
Memory                          Memory                          Current
  Modified       Improvement    Current        Modified          Current
    Estimated      Power
Instance                        Cell                            Read Access
  Read Access    Read Access    Read Access    Read Access       Power
    Power          Savings
Name                            Name                            Rate(%)
  Rate(%)        Rate(%)        Toggle Rate    Toggle Rate       (W)
    (W)            (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
ram_inst/i_sram_tcbram          sram32x64_hscm4                 92.31
  46.15          46.15          0.007692       0.03846           0.000814
    0.0007956      1.839e-05
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
1
```

See Also

- report_memory_redundant_write_cycles
- set_memory_cell_info
- compute_metrics
- update_metrics

---

report_memory_redundant_write_cycles

Analyzes input RTL FSDB for memory cells and reports redundant write cycle power.

Syntax

string *report_memory_redundant_write_cycles*

```
[object_list]
```

Data Types

object_list	list

Arguments

`object_list`

This command expects user to provide memory instance name for analysis. If no instance is specified all memory instances will be analysed.

Feedback

Description

This command analyzes input RTL FSDB or VCD waveform and reports the redundant cycles of Write Enable signal. The command reads input RTL FSDB and uses cycle power analysis to estimate the time windows in which write address or write data is constant and data has been valid for clock cycle. So, *Write Enable* signal can be disabled for the subsequent clock cycles, thereby reducing memory instance power.

The command only works in the average power analysis mode.

The command dumps out an output FSDB file which shows original *Write Enable* signal as well as estimated *Write Enable* signal after redundancy analysis. Output FSDB also contains original input and output signals of the memory instances so user can validate and change RTL code.

The command reports current, modified and improvement in Write Access Rate of *Write Enable* signal. Write Access Rate is defined as Static Probability of *Write Enable* input as percentage. The command also dumps modified *Write Enable* Toggle Rate after redundancy analysis. This can be used to validate estimated power of the memory instance using estimated enable signal.

The command reports current memory instance power, estimated power using new enable signal and power savings.

User can use output FSDB and command output report to modify user's RTL and input FSDB for making architecture-level decisions and improving memory power.

Examples

The following example shows the output report of *report_memory_redundant_write_cycles* command.

```
pwr_shell> report_memory_redundant_write_cycles ram_inst1/i_sram_tcbram
**************************************
Report : report_memory_redundant_write_cycles
Design : wrapper_tcbram
Version: U-2022.12-SP3
Date   : Tue May 23 14:21:35
**************************************

Memory                        Memory                        Current
  Modified       Improvement    Current        Modified       Current
    Estimated      Power
Instance                      Cell                          Write Access
  Write Access   Write Access   Write Access   Write Access   Power
    Power          Savings
Name                          Name                          Rate(%)
  Rate(%)        Rate(%)        Toggle Rate    Toggle Rate    (W)
    (W)            (W)
```

Feedback

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
ram_inst1/i_sram_tcbram       sram32x64_hscm4                92.31
  61.54          30.77           0.007692       0.03846         0.0006678
    0.0006675      5.484e-06
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------------------------
```

See Also

- report_memory_redundant_read_cycles
- set_memory_cell_info
- compute_metrics
- update_metrics

---

report_min_period

Displays minimum-period check information about specified pins or ports.

Syntax

string *report_min_period*

```
[-all_violators]
[-significant_digits digits]
[-nosplit]
[-path_type format]
[-input_pins]
[-derate]
[-variation]
[-nets]
[-crosstalk_delta]
[-transition_time]
[-capacitance]
[-clocks clock_list]
[-sms_scenarios sms_scenarios_list]
[port_pin_list]
```

Data Types

```
digits               integer
format               string
sms_scenarios_list   collection
port_pin_list        list
```

Feedback

Arguments

`-all_violators`

Reports only violating minimum period checks.

`-significant_digits` *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which defaults to 2. Use this option if you want to override the default.

`-nosplit`

Prevents line splitting in the report.

`-path_type` *format*

Specifies the format of the path report and how the clock path is displayed. Allowed values are

- *summary* (the default) - Displays one line for each path that shows only the required period, actual period, and slack.
- *short* - Shows only startpoints and endpoints in the clock path.
- *full_clock* - Shows full clock paths.
- *full_clock_expanded* - Shows full clock paths including all master clocks of a generated clock.

`-input_pins`

Reports the input pins. The default is to show only output pins.

`-derate`

Reports the derating factors. The default is to show no derate factors. This also shows the derate factor specified on the required period if one has been specified by the *set_timing_derate* command. Note that this option applies only when the *-path_type* option is set to *full_clock_expanded*.

`-variation`

Includes parametric on-chip variation (POCV) information in the report in columns labeled "Mean" and "Sensit". POCV analysis is enabled by setting the *timing_pocvm_enable_analysis* variable to *true*.

`-nets`

Reports nets. The default is not to show nets.

`-crosstalk_delta`

Reports the annotated delta delay and delta transition time. The deltas are computed during crosstalk signal integrity analysis, or they can be annotated manually using the *set_annotated_delay -delta_only* and *set_annotated_transition -delta_only* commands. Note that the *-crosstalk_delta* option only reports the calculated or annotated deltas; it does not initiate crosstalk analysis. Only deltas on input pins are shown. Delta transition time is shown only with the *-transition_time* option. The *-crosstalk_delta* option automatically sets the *-input_pins* option.

`-transition_time`

Reports the transition time (slew). The default is not to show transition time. For each driver pin or load pin the transition time is displayed in a column preceding incremental path delay.

`-capacitance`

Reports the total (lump) capacitance. The default is not to show capacitance. When the *-nets* option is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins.

`-clocks` *`clock_list`*

Reports minimum pulse width checks only from the specified clocks.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only min period violations compatible to the specified SMS scenarios collection are reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

`port_pin_list`

Reports the specified list of pins or ports. If you do not use this option, the report contains all pins and ports in the current design.

Description

The *report_min_period* command displays the following information for the minimum period check: required period, actual period, and by how much the constraint is violated or met (slack). This information is listed in order of decreasing slack starting with the worst violator.

If the *timing_remove_clock_reconvergence_pessimism* variable is set to *true*, clock reconvergence pessimism is removed from the slack on the reports.

To show more detailed minimum-period calculations, use the *-path_type* and *-input_pins* options.

When used under simultaneous multivoltage analysis (SMVA) analysis mode it will report only the cross-domain paths for minimum period checks.

You can also report minimum-period violations by using the *report_constraint* command with the *-min_period* option.

Examples

The following example generates a report of all minimum-period constraints in the current design.

```
pt_shell> report_min_period
****************************************
Report : min period
         -path_type summary
 ...
****************************************

   sequential_clock_min_period

                             Required        Actual
   Pin                     min period     min period        Slack
   -------------------------------------------------------------------
   b1/dst/CK (CK1 rise)         2.90           1.64          -1.26
 (VIOLATED)
   b1/dst/CK (CK1 fall)         2.90           1.80          -1.10
 (VIOLATED)
   b2/dst/CK (CK2 rise)         2.90           3.60           0.70   (MET)
   b2/dst/CK (CK2 fall)         2.90           3.80           0.90   (MET)
```

The following example generates a report of the minimum-period violation including the full clock path trace and the derate factor for the block1/test/CK pin.

```
pt_shell> report_min_period -path_type full_clock_expanded -derate
 block1/test/CK
****************************************
Report : min period
         -path_type full_clock_expanded
         -derate
 ...
****************************************

  Pin: block1/test/CK
  Related clock: CK2
  Check: sequential_clock_min_period

  Point                                   Derate    Incr       Path
  -----------------------------------------------------------------
  clock CK2 (fall edge)                             9.00       9.00
  clock source latency                              0.00       9.00
  CK2 (in)                                          0.00       9.00 f
  u2/Z (IBUF1)                             1.00     0.87 H     9.87 f
```

```
  block2/u1/Z (IBUF1)                     1.00      0.97 *    10.83 f
  block2/u2/Z (IBUF1)                     1.00      1.78 *    12.61 f
  block2/test/CK (NT9P32K1PG)             1.00      1.50 *    14.11 f
  open edge clock latency                                     14.11

  clock CK2 (fall edge)                            12.00      12.00
  clock source latency                              0.00      12.00
  CK2 (in)                                          0.00      12.00 f
  u2/Z (IBUF1)                            1.00      0.77 H    12.77 f
  block2/u1/Z (IBUF1)                     1.00      0.87 *    13.63 f
  block2/u2/Z (IBUF1)                     1.00      1.68 *    15.31 f
  block2/test/CK (NT9P32K1PG)             1.00      1.50 *    16.81 f
  clock uncertainty                                -0.20      16.61
  close edge clock latency                                    16.61
  ------------------------------------------------------------------
  required min period                     1.00                2.90 *
  actual period                                               2.50
  ------------------------------------------------------------------
  slack (VIOLATED)                                           -0.40
```

See Also

- report_constraint
- report_min_pulse_width
- report_timing
- report_default_significant_digits
- timing_remove_clock_reconvergence_pessimism

---

report_min_pulse_width

Displays minimum pulse width check information about specified pins or ports.

Syntax

string *report_min_pulse_width*

```
[-all_violators]
[-significant_digits digits]
[-nosplit]
[-path_type format]
[-input_pins]
[-derate]
[-variation]
[-nets]
[-crosstalk_delta]
[-transition_time]
[-capacitance]
[-voltage_swing]
```

Feedback

```
[-clocks clock_list]
[-sms_scenarios sms_scenarios_list]
[port_pin_list]
```

Data Types

digits	integer
format	string
sms_scenarios_list	collection
port_pin_list	list

Arguments

`-all_violators`

Reports only violating minimum pulse width checks.

`-significant_digits` *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which defaults to 2. Use this option if you want to override the default.

`-nosplit`

Prevents line splitting in the report.

`-path_type` *format*

Specifies the format of the path report and how the clock path is displayed. Allowed values are:

- *summary* (the default) - Shows one line for each path and only the required pulse width, actual pulse width, and slack
- *short* - Shows only startpoints and endpoints in the clock path
- *full_clock* - Shows full clock paths
- *full_clock_expanded* - Shows full clock paths including all master clocks of a generated clock

`-input_pins`

Indicates that input pins are shown in the path report. The default is to show only output pins.

`-derate`

Indicates that derate factors are shown in the path report. The default is to show no derate factors. This also shows the derate factor specified on the required pulse width if one has been specified by the *set_timing_derate*

command. Note that this option applies only when the *-path_type* option is set to *full_clock_expanded*.

`-variation`

Includes parametric on-chip variation (POCV) information in the report in columns labeled "Mean" and "Sensit". POCV analysis is enabled by setting the *timing_pocvm_enable_analysis* variable to *true*.

`-nets`

Indicates that nets are to be shown in the path report. The default is not to show nets.

`-crosstalk_delta`

Reports the annotated delta delay and delta transition time. The deltas are computed during crosstalk signal integrity analysis, or they can be annotated manually using the *set_annotated_delay -delta_only* and *set_annotated_transition -delta_only* commands. Note that the *-crosstalk_delta* option only reports the calculated or annotated deltas; it does not initiate crosstalk analysis. Only deltas on input pins are shown. Delta transition time is shown only with the *-transition_time* option. The *-crosstalk_delta* option automatically sets the *-input_pins* option.

`-transition_time`

Shows the transition time (slew). The default is not to show transition time. For each driver pin or load pin the transition time is displayed in a column preceding incremental path delay.

`-capacitance`

Shows the total (lump) capacitance. The default is not to show capacitance. When the *-nets* option is specified, the capacitance is printed on the lines with nets instead of the lines with driver pins.

`-voltage_swing`

Specifies that voltage swing check is performed in advanced waveform propagation mode. When computing the swing check, the minimum required width is set to 0.0 and the width measurement is made at a specified fraction of the supply voltage, which is specified by the variable *timing_voltage_swing_high_pulse_threshold* for the high pulse voltage swing computation and by the variable *timing_voltage_swing_low_pulse_threshold* for the low pulse voltage swing computation. Please note that voltage swing analysis should be enabled through the variable *timing_enable_voltage_swing* before invoking the command.

`-clocks` *`clock_list`*

Reports minimum pulse width checks only from the specified clocks.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only min pulse violations compatible to the specified SMS scenarios collection are reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

`port_pin_list`

Reports only the specified list of pins or ports. If you do not use this option, the report shows all pins and ports in the current design.

Description

The *report_min_pulse_width* command displays the following information for the minimum pulse width check: required pulse width, actual pulse width, and by how much the constraint is violated or met (slack). This information is listed in order of decreasing slack starting with the worst violator.

In minimum pulse width checking, by default, the tools measures the width of a pulse where the signal waveform crosses the voltage threshold at 50 percent of the supply voltage. If the *timing_enable_min_pulse_width_waveform_adjustment* variable is set to true, the measurement thresholds can be set to any fraction of the supply voltage between 0.0 and 1.0, through the variables *timing_min_pulse_width_high_pulse_threshold* and *timing_min_pulse_width_low_pulse_threshold* for the high pulse and the low pulse respectively.

If the *timing_remove_clock_reconvergence_pessimism* variable is set to true, clock reconvergence pessimism is removed from the slack on the reports.

More detailed minimum pulse width calculations are shown using the *-path_type* and *-input_pins* options.

Minimum pulse width violations can also be reported with the *report_constraint* command using the *-min_pulse_width* option.

Two types of minimum pulse width checks are performed:

- Clock Pulse Width Check at Sequential Devices: This check is performed on clock pins of sequential elements. This check verifies that a minimum separation exists between the trailing edge and leading edge of the clock that is clocking the sequential elements. The library defines the constraints, which are environmentally scalable. The most restrictive value of pulse width (library-specified or user-asserted) is used. No default value is assumed.
- Clock Tree Pulse Width Check at Logic Circuits: The clock tree pulse width check ensures that the clock pulse is propagated reliably through the logic. This check is performed on the combinational logic circuits through which the clock signals are propagated. No default is assumed for the clock tree pulse width check. The most restrictive value of pulse width (library-specified or user-asserted) is used.

Feedback

Note that if the *timing_enable_pulse_clock_constraints* variable is set to *true* (default value), the *report_min_pulse_width* command does not report pulse width checks in the pulse clock networks. To report pulse width checks in the pulse clock network use the *report_pulse_clock_min_width* command. To check pulse width in the pulse clock network using the *report_min_pulse_width* command, set the *timing_enable_pulse_clock_constraints* variable to *false*.

When used under simultaneous multivoltage analysis (SMVA) analysis mode it will report only the cross-domain paths for minimum pulse width checks.

Examples

The following example generates a report of all minimum pulse width violations in the current design.

```
pt_shell> report_min_pulse_width
****************************************
Report : min pulse width
         -path_type summary
 ...
****************************************

   sequential_clock_pulse_width

                  Required       Actual
   Pin            pulse width    pulse width      Slack
   ------------------------------------------------------------------
   ff1/CP           10.20          10.00          -0.20  (VIOLATED)
   ff2/CP           10.20          10.04          -0.16  (VIOLATED)
   ff3/CP            2.10           2.00          -0.10  (VIOLATED)
   ff3/CP            2.10           2.00          -0.10  (VIOLATED)
   ff2/CP           10.00           9.96          -0.04  (VIOLATED)


   clock_tree_pulse_width

                  Required       Actual
   Pin            pulse width    pulse width      Slack
   ------------------------------------------------------------------
   nand1/Z          10.00           9.58          -0.42  (VIOLATED)
   or1/B            10.00           9.58          -0.42  (VIOLATED)
   nand1/A          10.20          10.00          -0.20  (VIOLATED)
   or1/Z            10.20          10.04          -0.16  (VIOLATED)
   or1/Z            10.00           9.96          -0.04  (VIOLATED)
```

See Also

- remove_min_pulse_width
- report_constraint

Feedback

- report_pulse_clock_min_width
- report_timing
- set_min_pulse_width
- set_pulse_clock_min_width
- report_default_significant_digits
- timing_remove_clock_reconvergence_pessimism

---

report_mode

Displays a report of modes for specified cells or design.

Syntax

string *report_mode*

```
[-type cell | design]
[-nosplit]
[instance_list]
```

Data Types

instance_list	list

Arguments

`-type design | cell`

Indicates the type of mode to be reported. This option has the mutually exclusive valid values of either *design* or *cell*.

The *cell* value specifies that cell modes are to be reported. Cell modes are defined on library cells in the library.

The *design* value specifies that design modes are to be reported. Design modes are user-defined using the *define_design_mode_group* and *map_design_mode* commands. If the *-type* option is ommitted from the command, this is equivalent to specifying *-type cell*.

`-nosplit`

Most of the mode information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

instance_list

Specifies a list of instances whose modes are to be reported. By default, the report includes all cells that have modes. This option is only valid if -type has a cell value

Description

This command reports cell modes or design modes. When reporting cell modes, the report specifies whether the cell mode is enabled or disabled, and it also specifies the reason why the mode is enabled or disabled. There are four possible reasons as to why a mode can be enabled or disabled. They are

- cell - Indicates that the cell mode has been set directly using the *set_mode -type cell* command.
- design - Indicates that the cell mode has been set due to the activation of a design mode using the *set_mode -type design* command. The design mode must have previously been mapped to the cell mode using the *map_design_mode* command. The report also specifies the name of the activated design mode. To view more details about the design mode, use the *report_mode -type_design* command.
- cond - Indicates that the cell mode has been set due to evaluation of the mode condition during constant propagation.
- default - Indicates that the cell mode has not been set by any of the above reasons.

When reporting design modes the report displays the design modes specified on the current design and for each design mode, reports the cell modes and moded pins associated with that design mode.

Examples

The following example reports cell mode information for the design top_design. Two cells have modes, Uram1 and Uram2 are instances of the library cell RAM2_core. rw is a cell mode group defined on the library cell RAM2_core. This cell mode group has two cell modes read and write. No modes are currently active so all modes are enabled and the reason is given as default.

```
pt_shell> report_mode

****************************************
Report : mode
Design : top_design
****************************************

Cell                              Mode(Group)        Status      Reason
--------------------------------------------------------------------------------
---
Uram1/core(RAM2_core)             read(rw)           ENABLED     default
                                  write(rw)          ENABLED     default
```

```
------------------------------------------------------------------------
---
Uram2/core(RAM2_core)              read(rw)          ENABLED     default
                                   write(rw)         ENABLED     default
------------------------------------------------------------------------
---
```

The following example shows a report of modes specified for the two RAMs that have modes in the design. Ram Uram1 is set in mode read, and Ram Uram2 is set in mode write. Thus, all timing arcs of RAM Uram1 associated with mode read are enabled , and all timing arcs associated with mode write are disabled.

```
pt_shell> set_mode read Uram1/core
pt_shell> set_mode write Uram2/core
pt_shell> report_mode

****************************************
Report : mode
Design : top_design
****************************************

Cell                               Mode(Group)     Status          Reason
------------------------------------------------------------------------
---
Uram1/core(RAM2_core)              read(rw)        ENABLED         cell
                                   write(rw)       disabled        cell
------------------------------------------------------------------------
---
Uram2/core(RAM2_core)              read(rw)        disabled        cell
                                   write(rw)       ENABLED         cell
------------------------------------------------------------------------
---
```

The following example reports that two design modes have been specified using the *set_mode -type design* command. For each design mode, it reports any mapping of cell modes, as specified by the *map_design_mode* command. The report also displays any paths that are design mode dependent, as specified by the *map_design_mode -from* or *map_design_mode -to* command.

```
pt_shell> map_design_mode  {read,latching} {U2/core,U1/core} read
pt_shell> map_design_mode  -from {INFF1/CLK} -to {INFF3/D}
pt_shell> map_design_mode  -from {INFF1/CLK} -to {INFF4/D}
pt_shell> map_design_mode  {write,transparent} {U2/core,U1/core} write
pt_shell> report_mode -type design


****************************************
Report : design_mode
Design : misc
Version: v4.0a-development
Date   : Fri Mar 29 13:29:05 1996
```

```
****************************************

--------------------------------------------------------------
Design Mode Group : 'default'
      Design Mode : 'read' (is current design mode)
      ------------------------------------------------------------
         Cell                          Mode(Group)
         ------------------------------------------------------------
         U2/core                       latching(output_latch)
                                       read(rw)
         ------------------------------------------------------------
         U1/core                       latching(output_latch)
                                       read(rw)
         From Pin
         ------------------------------------------------------------
         INFF1/CLK
         INFF1/CLK

         To Pin
         ------------------------------------------------------------
         INFF3/D
         INFF4/D
--------------------------------------------------------------

--------------------------------------------------------------
Design Mode Group : 'default'
      Design Mode : 'write'
      ------------------------------------------------------------
         Cell                          Mode(Config)
         ------------------------------------------------------------
         U2/core                       transparent(output_latch)
                                       write(rw)
         ------------------------------------------------------------
         U1/core                       transparent(output_latch)
                                       write(rw)
--------------------------------------------------------------
```

See Also

- map_design_mode
- define_design_mode_group
- remove_design_mode
- reset_mode
- set_mode

Feedback

report_multi_input_switching_coefficient

Reports delay coefficient for multi-input switching (MIS) analysis.

Syntax

int *report_multi_input_switching_coefficient*

```
[-nosplit]
[-significant_digits num_digits]
[object_list]
```

Data Types

num_digits	integer
object_list	list

Arguments

`-nosplit`

Does not split lines in report when columns overflow.

`-significant_digits num_digits`

Specifies the number of digits displayed in the report.

object_list

Specifies a list of library cells.

Description

This command reports delay coefficient for multi-input switching (MIS) analysis for the specified list of library cells. If you do not specify a list of library cells, the tool reports the coefficients for all library cells.

Examples

The following example reports all coefficients set by set_multi_input_switching_coefficient command:

```
pt_shell> report_multi_input_switching_coefficient
```

See Also

- set_multi_input_switching_coefficient
- reset_multi_input_switching_coefficient
- si_enable_multi_input_switching_analysis
- si_enable_multi_input_switching_timing_window_filter

Feedback

report_multi_input_switching_lib_cells

Reports candidate library cells for advanced multi-input switching (MIS) analysis.

Syntax

int *report_multi_input_switching_lib_cells*

Description

Advanced multi-input switching (MIS) analyzes library cell data, along with actual pin slew, arrival, and waveform information, to estimate multi-input switching (speedup) effects. See the *si_multi_input_switching_analysis_mode* man page for details.

User-defined coefficients are not required, but the following library data is required:

- CCS timing models
- CCS noise models
- All *when* conditions characterized

The following library cell types are supported for advanced MIS analysis:

```
AND2   OR2   NAND2   NOR2   AO211   AOI211   OA211   OAI211
AND3   OR3   NAND3   NOR3   AO21    AOI21    OA21    OAI21
AND4   OR4   NAND4   NOR4   AO22    AOI22    OA22    OAI22
                            AO31    AOI31    OA31    OAI31
```

This command reports the list of library cells that qualify for advanced MIS calculation.

Examples

The following example reports all the library cells that are candidates for advanced MIS analysis:

```
pt_shell> report_multi_input_switching_lib_cells
```

report_multi_scenario_design

Creates a detailed report on user defined multi-scenario objects and attributes.

Syntax

status *report_multi_scenario_design*

```
[-scenario]
[-session]
[-license]
```

Feedback

Arguments

`-scenario`

Reports in detail on the current scenarios that have been defined. Details reported include the name of the scenario, the files used to generate the image for the scenario (i.e. common and specific data files), the scenario used to generate the baseline image for the scenario, the location of the current image, the focus of the scenario and the affinity specification for the scenario. For detailed description of the settings of the focus field, see the description section below.

`-session`

Reports in detail on the current session. This option reports the number of scenarios defined in the current session. Three fields are reported for each scenario in the current session: the command focus of the scenario, the name of the scenario, and the scenario which provides the image to the scenario.

`-license`

Reports in detail on license usage and limits set.

Description

This command is available only if you invoke the pt_shell with the *-multi_scenario* option.

The *report_multi_scenario_design* command generates a report on user-defined multi-scenario analysis. The level of detail reported depends on the options specified. By selecting no options, the *report_multi_scenario_design* command issues a short summary of current user-defined objects and settings. The summary reports on whether or not a session has been defined. The number of defined scenarios and licenses in use are also reported. The report generated can report in detail on the following topics: scenarios, sessions, and licenses.

The *-scenario* option generates a report on all the scenarios that exist in the manager process. The report shows several pieces of information about each scenario:

- The name of the scenario
- The files used to generate the images for the scenario (i.e. common and specific data files)
- The scenario used to generate the baseline image for the scenario
- The location of the scenario's current image
- The focus of the scenario, which can be one of three possible values

```
-"Out of focus" means the scenario is not in the current_session.
 -"In current session" means the scenario is in the current session
 but
```

```
     it is not in command focus. i.e. It will not receive commands from
    the
     manager.
    -"In current session and command focus" means the scenario is in the
     current session and in command focus. i.e. It will receive commands
     from the manager.
```

- The affinity of the scenario

See the *current_scenario* command for changing command focus.

The *-session* option reports the scenarios defined in the current session. Along with the number of scenarios, some scenario-specific data is reported. The focus field of the scenario shows whether the scenario is in the session (Session), or whether it is also in command focus (Command). Also reported is the scenario which provides the image to each scenario.

The *-license* option shows the numbers of licenses checked out for each feature and the user defined limits for each feature.

Examples

In the following example, several scenarios are created and the current session is set. Then the *report_multi_scenario_design* command is used to examine the multi scenario information available to the manager.

```
pt_shell>create_scenario -name scen1 -common_data {com_s1.tcl} \\
  -specific_data {spec_s1.tcl spec_s2.tcl spec_s3.tcl}
1
pt_shell>create_scenario -name scen2 -common_data {com_s2.tcl}
 -specific_data {spec_s2.tcl}
1
pt_shell>create_scenario -name scen3 -common_data {com_s3.tcl}
 -specific_data {spec_s3.tcl}
1
pt_shell>create_scenario -name scen4 -common_data {com_s3.tcl}
 -specific_data {spec_s3.tcl}
1
pt_shell> current_session {scen1 scen2 scen3}
1
pt_shell> report_multi_scenario_design
1


****************************************
Report : multi_scenario_design
Version: W-2004.12-Beta3
Date   : Mon Nov  8 08:59:13 2004
****************************************

Summary
```

Feedback

```
-------------------------------------

Current session                       :  Defined
Number of defined scenarios           :  4
Number of scenarios in command focus  :  3

1
```

See Also

- create_scenario
- current_session
- current_scenario

---

report_multi_user_server

Reports the configuration of the current multi-user server session.

Syntax

status *report_multi_user_server*

Description

The *report_multi_user* command reports information about the current multi-user server session, including all clients that are active.

It reports the following detailed information:

- Server-related information
  - Server Name - The name of the server session.
  - Session ID - A string which uniquely identifies the multi-user session.
  - Server Log - The name of the file which will contain information related to the active multi-user session.
  - Logging Level - Verbose level of information written in log file. Can be changed using the *multi_user_logging_level* variable.
  - Access Group - Name of the UNIX group to restrict access.
  - Client Cores - Number of cores available for use by the multi-user clients.
  - Client Limit - Maximum number of clients allowed in the current session.
  - Active Clients - Number of clients currently active.
- Client-related information

- Client Name - The name of the client session.
- PID - Process ID of the client session.
- Display - DISPLAY environment used by client to run design in interactive mode
- Script File - Name of script file containing commands which will be executed by clients.
- Output Log - File containing output of commands executed by the client.

The *report_multi_user_server* command is supported only in a multi-user server session.

Examples

In the following example, a multi-user server is running with 2 cores, and a client is connected and running in non-interactive mode:

```
pt_shell> report_multi_user_server
  ****************************************
  Report : Multi-User Report
  ****************************************

  Report on current server configuration:
  ---------------------------------------
    Server Name:     user.123
    Session ID:      30:35787
    Server Log:      user.123_multi_user.log
    Logging Level:   low
    Access Group:    synopsys
    Client Cores:    2 (Total)
    Client Limit:    2
    Active Clients:  1

    Report on active clients:
    -----------------------------
    Client Name    PID    Display           Script file      Output log

 ------------------------------------------------------------------------
---
    c1.8748        8748   N/A               ./script_file           ./log
```

See Also

- start_multi_user_client
- start_multi_user_server
- stop_multi_user_server
- stop_multi_user_client

Feedback

report_name_mapping

Reports the user-defined name-mapping rules.

Syntax

string *report_name_mapping*

```
[-nosplit]
```

Arguments

`-nosplit`

Prevents the line splitting if column overflows.

Description

The *report_name_mapping* command reports the user-defined name-mapping rules you have set using the *set_rtl_to_gate_name* command.

Examples

In the following example, all the user-defined name-mapping rules are reported.

```
pt_shell> set_rtl_to_gate_name -rtl a_net -gate n272
1
pt_shell> set_rtl_to_gate_name -rtl a_pin -gate U3/D0
1
pt_shell> set_rtl_to_gate_name -rtl a_cell -gate U3
1
pt_shell> report_name_mapping

****************************************
Report : report_name_mapping
Design : mac
Version: D-2010.06-Alpha2
Date   : Fri Apr  2 11:12:53 2010
****************************************

      Attributes
      ----------
       v - Inverted

------------------------------------------------------------------------
-------
RTL Name                      Gate Level Name               Type
 Attributes
------------------------------------------------------------------------
-------
a_cell                        U3                            cell
a_pin                         U3/A                          pin
a_net                         n39                           net
```

Feedback

```
-------------------------------------------------------------------------
-------
```

See Also

- set_rtl_to_gate_name

---

report_net

Generates a report of net information.

Syntax

string *report_net*

```
[-connections]
[-verbose]
[-significant_digits digits]
[-segments]
[-nosplit]
[net_names]
```

Data Types

digits	int
net_names	list

Arguments

`-connections`

Indicates that the report shows net connection information.

`-verbose`

Indicates that the report shows verbose connection information. This must be used with the *-connections* option.

`-significant_digits digits`

Specifies the number of digits to the right of the decimal point that are reported. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which the default value is 2. Use this option if you want to override the default. Fixed-precision floating-point arithmetics is used for delay calculation; therefore, the actual precision might depend on the platform. For example, on SVR4 UNIX see FLT_DIG in limits.h, the upper bound on a meaningful value of the argument to the *-significant_digits* option is then FLT_DIG - ceil(log10(fabs(*any_reported_value*))).

`-segments`

Indicates that the report shows all global segments for each net requested. Global net segments are all those physically connected across all hierarchical boundaries.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

net_names

Specifies a list of nets that is reported. The default is to report all nets in the current instance or current design.

Description

The command displays information about the nets in the design of the current instance, or in the current design. If the *current_instance* command is set, the report generates for the design of that instance; otherwise, the report generates for the current design. Attributes, such as annotated resistance and annotated capacitance, are displayed.

If the *-connections* optin is used, PrimeTime lists the leaf cell pins connected to each net.

Examples

The following example reports all nets in the design.

```
pt_shell> report_net

****************************************
Report : net
Design : top
Version: 1997.01-development
Date   : Wed Aug  7 09:28:05 1996
****************************************

Attributes:
    c - annotated capacitance
    r - annotated resistance

Net                  Fanout     Fanin       Cap   Resistance    Pins
 Attributes
-------------------------------------------------------------------------
-------
a                         2         1      2.00         0.00       3
b                         1         1      1.00         0.00       2
c                         1         1      1.00         0.00       2
d                         1         1      1.00         0.00       2
e                         1         1      1.00         0.00       2
```

```
en                        20          1      25.00          0.00       21
f                          1          1       1.00          0.00        2
g                          1          1       1.00          0.00        2
h                          1          1       1.00          0.00        2
i1                         1          1       1.00          0.00        2
i2                         1          1       1.00          0.00        2
i3                         1          1       1.00          0.00        2
i4                         1          1       1.00          0.00        2
j                          1          1       1.00          0.00        2
k                          1          1       1.00          0.00        2
l                          1          1       1.00          0.00        2
m                          2          1       2.00          0.00        3
n                          1          1       1.00          0.00        2
o                          1          1       1.00          0.00        2
o1                         1          1       0.00          0.00        2
o2                         1          1       0.00          0.00        2
p                          1          1       1.00          0.00        2
q                          2          1       2.00          0.00        3
r                          1          1       1.00          0.00        2
s                          1          1       1.00          0.00        2
t                          1          1       1.00          0.00        2
u                          1          1       1.00          0.00        2
w                          1          1       1.00          0.00        2
y                          1          1       1.00          0.00        2
z                          1          1       2.00          0.00        2
-------------------------------------------------------------------------
-------
Total 30 nets             52         30      56.00          0.00       82
Maximum                   20          1      25.00          0.00       21
Average                 1.73       1.00       1.87          0.00     2.73
```

The following example displays a verbose connection report for net z.

```
pt_shell> report_net -verbose -connections z

****************************************
Report : net
        -connections
        -verbose
Design : top
Version: 1997.01-development
Date   : Wed Aug  7 09:30:38 1996
****************************************


Connections for net 'z':
    pin capacitance:    2
    wire capacitance:   0
    total capacitance:  2
    wire resistance:    0
    number of drivers:  1
    number of loads:    1
    number of pins:     2
```

```
Driver Pins             Type                Pin Cap
------------            ----------------    --------
z/Z                     Output Pin (NOR2)    0

Load Pins               Type                Pin Cap
------------            ----------------    --------
o2/B                    Input Pin (BUF)      2
```

See Also

- current_design
- current_instance
- report_cell
- report_design

---

report_net_connectivity

Reports the ports, instances, and cells connected to the specified nets.

Syntax

status *report_net_connectivity*

```
        nets
```

Data Types

nets list or collection

Arguments

nets

A list of one or more nets for which to report net connectivity. At least one net is required. Glob-style wildcards (*) are supported.

Description

This command reports the ports, instances, and cells connected to the specified nets.

Examples

The following example shows a detailed connectivity report for nets with escape characters.

Feedback

```
pt_shell> report_net_connectivity [get_nets -exact {net\\[0\\]}]
============================
Net: count_en
Report Type: Net Connectivity
============================
--------   ---------   ------------   ------------
*P Name    Direction   x-coordinate   y-coordinate
--------   ---------   ------------   ------------
count_en   in          492400.000000  400.000000
--------   ----------- ----   ------------  -----------
*I Name    Direction   Cell   x-coordinate  y-coordinate
--------   ---------   ----   ------------  -----------
U86/A      in          U86    342000.000000 254600.000000
U87/A      in          U87    319600.000000 254000.000000
---- ---------------- ---------------- ---------------- ----------------
Cell x-coordinate min y-coordinate min x-coordinate max  y-coordinate max
---- ---------------- ---------------- ---------------- ----------------
U86  342000.00         254600.00         345200.00         257200.00
U87  312600.00         254000.00         319600.00         258950.00
```

Where,

*P report has pin names, direction, and x and y coordinates.

*I report has port names, direction, cell name, and x and y coordinates.

Cell report has cell names, and x and y coordinates of bounding box.

---

report_noise

Reports noise analysis information.

Syntax

int *report_noise*

```
[-above]
[-below]
[-low]
[-high]
[-nworst_pins pin_count]
[-significant_digits digits]
[-verbose]
[-nosplit]
[-slack_lesser_than slack_limit]
[-slack_type height | area | area_percent]
[-all_violators]
[-data_pins]
[-clock_pins]
[-async_pins]
[-sms_scenarios list]
[object_list]
```

Feedback

Data Types

digits	integer
object_list	list
pin_count	integer
slack_limit	float

Arguments

`-above`

Performs the reporting only above the rails. If this option is combined with the *-low* option, it reports the noise bumps above the low rail. If it is combined with the *-high* option, it reports the noise bumps above the high rail. Otherwise, it reports the noise bumps above either rail.

`-below`

Performs the reporting only below the rails. If this option is combined with the *-low* option, it reports the noise bumps below the low rail. If it is combined with the *-high* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps below either rail.

`-low`

Performs the reporting only for the low rail. If this option is combined with the *-above* option, it reports the noise bumps above the low rail. If it is combined with the *-below* option, it reports the noise bumps below the low rail. Otherwise, it reports the noise bumps both below and above the low rail.

`-high`

Performs the reporting only for the high rail. If this option is combined with the *-above* option, it reports the noise bumps above the high rail. If it is combined with the *-below* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps both below and above the high rail.

`-nworst_pins` *pin_count*

Specifies the number of load pins to be reported. Values of 1 or greater are allowed. The default is 1.

`-significant_digits` *digits*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default value is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

`-verbose`

Shows more details about the calculation of total noise on each load pin, including the individual contribution of each aggressor as well as noise bumps propagated from previous logic stages in the design.

`-nosplit`

If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

`-slack_lesser_than` *`slack_limit`*

Indicates that only those pins with a slack less (more negative) than *slack_limit* are to be shown.

`-slack_type area | height | area_percent`

Specifies the type of slack to be used. Valid values are *area*, *height*, and *area_percent*.

The voltage margin is defined by the noise bump height and noise immunity curves or DC noise margin. A *-slack_type* value of *height* reports noise slack as the voltage margin. A *-slack_type* value of *area* reports noise slack as the voltage margin multiplied by the noise bump width. A *-slack_type* value of *area_percent* reports noise slack as the percentage of the noise constraint area. The noise constraint area is computed by multiplying the noise height constraint by the noise bump width.

The default is *height*.

`-all_violators`

Indicates that only violating pins (negative slack) are to be shown. This option cannot be used with the *-slack_lesser_than* option. If this option is used with the *-nworst_pins* option, the number of violating pins will be limited by that value.

`-data_pins`

Indicates that the reporting is to be performed only for data pins of sequential cells. This is similar to passing the results of the *all_registers -data_pins* command to the *report_noise* command.

`-clock_pins`

Indicates that the reporting is to be performed only for clock pins of sequential cells. This is similar to passing the results of the *all_registers -clock_pins* command to the *report_noise* command.

`-async_pins`

Indicates that the reporting is to be performed only for asynchronous pins of sequential cells. This is similar to passing the results of the *all_registers -async_pins* command to the *report_noise* command.

`-sms_scenarios` *`list`*

Specifies SMS scenarios for scenario specific reporting. By default, the worst-case result across all scenarios is reported.

`object_list`

Specifies the load pins for which noise reporting is performed. If no pins are specified, reporting is performed for the entire design.

Description

This command provides a report of noise information for the current design.

It reports the top *-nworst_pins* pins or the pins specified by the *object_list* argument in increasing order of their slack values. By default, the slack values used are based on *-slack_type height*. Using the *-slack_type area* option may result in a different order of pins reported. Pins with positive slack are not reported in any particular order.

By default, a summary report is generated that shows the total crosstalk noise bump height and width on the load pins. If the *-verbose* option is used, the individual contribution of each effective aggressor and the amount of propagated noise is also reported.

Examples

The following command generates a report for the worst total noise bump above the low rail in the current design.

```
pt_shell> report_noise -above -low
```

The following example generates a report for the five worst total noise bumps above the low rail in the current design.

```
pt_shell> report_noise -above -low -nworst_pins 5
```

The following example generates a report for the worst total noise bump above the low rail and also the worst total noise bump below the low rail in the current design.

```
pt_shell> report_noise -low
```

The following example generates a report for the worst total noise bump for each of the four cases of above the low rail, below the low rail, above the high rail, and below the high rail in the current design.

```
pt_shell> report_noise -low -high
```

Feedback

See Also

- update_noise
- report_default_significant_digits

---

report_noise_calculation

Reports the detailed calculation of noise information for the specified net arc.

Syntax

string *report_noise_calculation*

```
[-above]
[-below]
[-low]
[-high]
[-significant_digits digits]
[-nosplit]
-from from_pin
-to to_pin
[-sms_scenarios list]
```

Data Types

digits	integer
from_pin	list
to_pin	list

Arguments

`-above`

Specifies a noise calculation for the above ground or power rails noise analysis region.

`-below`

Specifies a noise calculation for the below ground or power rails noise analysis region.

`-low`

Specifies a noise calculation for ground rail noise.

`-high`

Specifies a noise calculation for power rail noise.

`-significant_digits digits`

Specifies the number of digits after the decimal point displayed for time values in the generated report. Allowed values are 0-13; the default is determined by

the *report_default_significant_digits* variable, which has a default of 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

`-from` *`from_pin`*

Specifies the startpoints of a net arc within a design. The *from_pin* value must be a driver pin or port.

`-to` *`to_pin`*

Specifies the endpoints of a net arc within a design. The *to_pin* value must be the load pin or port of the same net.

`-sms_scenarios` *`list`*

Specifies SMS scenarios for scenario specific reporting. By default, reporting is done for all scenarios.

Description

This command reports the detailed calculation of noise information for the specified net arc. This command uses the noise analysis settings for the design set by the *set_noise_parameters* command. If needed, it updates the noise information for the design.

The report shows some general information about the victim, as well as details of noise bump and noise slack calculations.

The general information section includes the victim driver net, pin, noise composite aggressor mode, and any applicable derating factors set by the *set_noise_derate* command.

The noise calculation section includes the total noise bump as well as the estimated noise bumps induced from each of the aggressors. Note that individual aggressor noise bumps are reported only to show their relative contributions. The linear summation of all individual noise bumps might not match the total noise bump due to the nonlinear behavior of the coupled clusters.

The aggressor attribute column shows more details about each of the effective aggressors. Aggressors which are filtered are not included in this report.

The following is the list of possible attributes an aggressor can have:

```
Attributes:
  A - aggressor is active
  C - aggressor is a composite aggressor
  D - aggressor is analyzed with detailed engine
  E - aggressor is screened due to user exclusion
  G - aggressor is analyzed with gate level simulator
  I - aggressor has infinite window
  L - aggressor is screened due to logical correlation
  S - aggressor is screened due to small bump height
  X - aggressor is screened due to aggressor exclusion
```

An active aggressor is an aggressor that contributes to the worst-case alignment scenario of all the aggressors.

A composite aggressor models the combined effect of a group of weak aggressors to improve runtime and memory capacity.

An aggressor can be screened from the analysis if it has no significant impact on the victim. For example, when the aggressor is excluded by using the *case_analysis* or *set_si_delay_analysis* commands, or if the bump height that it induces on the victim is very small.

An aggressor is analyzed using the CCS noise gate-level simulation engine if the amplitude of the total noise glitch is significant compared to the lowest noise immunity of the load cells. In this mode, all aggressor drivers as well as the victim driver are analyzed using CCS noise advanced engine and this attribute 'G' shows up on the total noise line.

An aggressor is infinite-window when it has no fixed timing relationship with the victim. For example, an aggressor driven by a clock asynchronous to the clock that drives the victim is infinite-window.

An aggressor has logical correlation if there is a common point in the transitive fanin of both aggressor and the victim, or two aggressors.

An aggressor can be excluded from the analysis by using the *case_analysis* or *set_si_delay_analysis* commands.

The noise slack calculation section shows the details of how the area slack and height slack are derived for the victim, and also what type of constraint was used to derive the slack. The height slack is the difference of the actual bump height versus the bump height which causes the failure with the same width. The area slack is the area that needs to be added to the actual bump to cause a noise failure.

Feedback

Examples

The following command reports the noise calculation for the driver pin buf2/ZN and load pin buf5/I for below the high rail noise region:

```
pt_shell> report_noise_calculation -below -high -from buf2/ZN -to buf5/I
 ...

Units:  1ns  1pF  1kOhm

Analysis mode                          : report_at_source
Region                                 : below_high
Victim driver pin                      : buf2/ZN
Victim driver library cell             : mylib/INVD3
Victim net                             : I2
Victim driver effective capacitance    : 0.200827

Steady state resistance source         : library set CCS noise iv curve
Driver voltage swing                   : 1.080000
Noise derate height offset             : 0.000000
Noise derate height scale factor       : 1.000000
Noise derate width scale factor        : 1.000000
Noise effort threshold                 : 0.000000
Noise composite aggressor mode         : disabled

Noise calculations:

Attributes:
  A - aggressor is active
  C - aggressor is a composite aggressor
  D - aggressor is analyzed with detailed engine
  E - aggressor is screened due to user exclusion
  G - aggressor is analyzed with gate level simulator
  I - aggressor has infinite window
  L - aggressor is screened due to logical correlation
  S - aggressor is screened due to small bump height
  X - aggressor is screened due to aggressor exclusion

                        Height    Width     Area      Aggressor
 Attributes
-------------------------------------------------------------------------
---
 Aggressors:
 I1                     0.300604  0.786466  0.118207  A I D
 I3                     0.300604  0.786466  0.118207  A I D
 Total:                 0.497124  0.919737  0.228612  G

Noise slack calculation:

Constraint type: library CCS noise immunity

                        Height              Area
-------------------------------------------------------------------
```

```
 Required                0.581570            (0.581570 * 0.919737)   -
 Actual                  0.497124            (0.497124 * 0.919737)
-------------------------------------------------------------------------
 Slack                   0.084445            0.077668
```

See Also

- report_noise
- set_noise_derate
- set_noise_parameters
- set_si_aggressor_exclusion
- update_noise
- si_noise_composite_aggr_mode

---

report_noise_parameters

Reports status of the noise analysis parameters for the current design.

Syntax

```
int report_noise_parameters
```

Arguments

None.

Description

This command reports status of the parameters that are considered during the noise analysis. If this command is performed, the settings of the parameters for the noise analysis are reported: beyond the rail analysis, arrival times of aggressors and propagation of noise.

Examples

The following example shows the report format:

```
pt_shell> report_noise_parameters
report_noise_parameters
****************************************
Report  : noise_parameters
Design  : top
Version : Y-2006.06-VA-STA-Beta1-DEV
Date    : Thu Jan 26 09:46:43 2006
****************************************

 ignore arrival       : true
```

```
include beyond Rails : true
enable propagation   : true
```

See Also

- update_noise
- report_noise
- set_noise_parameters
- reset_noise_parameters

---

report_noise_violation_sources

Reports noise violation sources for failing endpoints.

Syntax

int *report_noise_violation_sources*

```
[-above]
[-below]
[-low]
[-high]
[-nworst_endpoints pin_count]
[-max_sources_per_endpoint pin_count]
[-significant_digits digits]
[-verbose]
[-nosplit]
[-slack_type slack_type]
[object_list]
```

Data Types

pin_count	integer
digits	integer
slack_type	float
object_list	list

Arguments

`-above`

Performs the reporting only above the rails. If this option is combined with the *-low* option, it reports for the noise bumps above the low rail. If it is combined with the *-high* option, it reports the noise bumps above the high rail. Otherwise, it reports the noise bumps above the high rail and above the low rail.

`-below`

Performs the reporting only below the rails. If this option is combined with the *-low* option, it reports for the noise bumps below the low rail. If it is combined with the *-high* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps below the high rail and below the low rail.

`-low`

Performs the reporting only for the low rail. If this option is combined with the *-above* option, it reports the noise bumps above the low rail. If it is combined with the *-below* option, it reports the noise bumps below the low rail. Otherwise, it reports the noise bumps for both below and above the low rail.

`-high`

Performs the reporting only for the high rail. If this option is combined with the *-above* option, it reports the noise bumps above the high rail. If it is combined with the *-below* option, it reports the noise bumps below the high rail. Otherwise, it reports the noise bumps for both below and above the high rail.

`-nworst_endpoints` *`pin_count`*

Specifies the number of endpoint pins to be reported. Any number greater than 1 is accepted; the default value is 1.

`-max_sources_per_endpoint` *`pin_count`*

Specifies the number of violation source pins per endpoint to be reported. Any number greater than 1 is accepted; the default value is 1.

`-significant_digits` *`digits`*

Specifies the number of digits after the decimal point to be displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default value is 2. Use this option if you want to override the default. This option controls only the number of digits displayed, not the precision used internally for analysis. For analysis, PrimeTime uses the full precision of the platform's fixed-precision, floating-point arithmetic capability.

`-verbose`

Shows details about pins in fan-in cone of the logic from violation sources to endpoint. Pins are ordered from the endpoint to sources and levelized by the distance from the endpoint. The worst slacks are shown on the top.

`-nosplit`

If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-

splitting and facilitates writing software to extract information from the report output.

`-slack_type` *`slack_type`*

Specifies the type of slack to be used. Valid values are *area*, *height*, and *area_percent*. The voltage margin is defined by the noise bump height and noise immunity curves or DC noise margin. A *slack_type* value of *height* reports noise slack as the voltage margin. This setting is the default. A *slack_type* value of *area* reports noise slack as the voltage margin multiplied by the noise bump width. A *slack_type* value of *area_percent* reports noise slack as the percentage of the noise constraint area. The noise constraint area is computed by multiplying the noise height constraint by the noise bump width.

`object_list`

Specifies the load pins for which the noise reporting is performed. If no pin is specified, reporting is performed on the entire design.

Description

Provides a report of noise violation source information for failing noise endpoints of the current design.

A noise endpoint is one of the followings: - Data pin, clock pin, or asynchronous pin of flip-flops - Input pin of level sensitive latch - Output port - Load pin of gates where noise exceeds the threshold

A failing endpoint is an endpoint with negative slack value. A violation source is the source of violation that causes a failure in the endpoint.

This command reports the violation sources of the specified or the nworst endpoints.

By default, the non verbose report is generated which shows the total crosstalk noise bump height and width as well as slack on the load pins.

Beginning with the F-2011.06 release, the *-slack_type* option defaults to *height* instead of *area* since it is a more direct representation of the noise violation on a pin.

Examples

The following example generates a report for the worst violation source of the worst endpoint above the low rail in the current design.

```
pt_shell> report_noise_violation_sources -above -low
```

The following example generates a report for the worst violation source of five worst endpoints above the low rail in the current design.

```
pt_shell> report_noise_violation_sources -above -low -nworst_pins 5
```

Feedback

The following example generates a report for the worst violation source above the low rail and also the worst violation source below the low rail in the current design.

```
pt_shell> report_noise -low
```

The following example generates a report for the worst violation source for each of the four cases of above the low rail, below the low rail, above the high rail, and below the high rail in the current design.

```
pt_shell> report_noise -low -high
```

See Also

- update_noise
- set_noise_parameters
- report_noise
- report_default_significant_digits

---

report_nonphysical_resistors

Reports the nonphysical resistors associated with the specified nets.

Syntax

status *report_nonphysical_resistors*

```
[-verbose]
[-of_objects nets]
[-from node1 -to node2]
```

Data Types

nets	list
node1	string
node2	string

Arguments

`-verbose`

Provides additional information about the coupling capacitors.

`-of_objects nets`

A list of one or more nets for which to report the nonphysical resistors. At least one net is required. Glob-style wildcards (*) are supported.

This option is mutually exclusive with the *-from* and *-to* options.

`-from` *node1*

Specifies a pin, port, or net internal node. The tool reports the nonphysical resistors for paths between this node and the node specified in the *-to* option. Both nodes must both belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

`-to` *node2*

Specifies a pin, port, or net internal node. The tool reports the nonphysical resistors for paths between this node and the node specified in the *-from* option. Both nodes must both belong to the same net.

This option is mutually exclusive with the *-of_objects* option.

Description

This command reports the nonphysical resistors associated with the specified nets.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a default nonphysical resistor report.

```
pt_shell> report_nonphysical_resistors -of_objects min_lsb[5]

****************************************
Report : Non-Physical Resistors Net Based summary
Design : toprt
Version: Q-2019.12
Date   : Mon Nov 11 17:14:53 2019
****************************************

Net: min_lsb[5]

Node1           Node2           Resistance
=====           =====           ==========
min_lsb/U24/X   min_lsb[5]:24   0.000001
```

The following example shows a verbose nonphysical resistor report.

```
pt_shell> report_nonphysical_resistors -of_objects min_lsb[5] -verbose

****************************************
Report : Non-Physical Resistors Net Based detailed
Design : toprt
Version: Q-2019.12
Date   : Mon Nov 11 17:15:12 2019
****************************************

Non-Physical Resistor Categories:
A - To connect resistively connected groups (RCGs) when physical opens
```

```
    exist in the design
B - To connect electrically equivalent nodes under specific situations
  - To short bulk nodes of MOS devices
  - Superconductive metal resistors
  - To short overlapping skip cell material
  - Very small aspect ratio resistors (l<<w)
C - Shorting resistors used on device layers in a special transistor
    level flow
D - To short pin shapes that are not explicitly connected together
E - Superconductive via resistors
F - Gate adjustment resistors (Rgdelta)
G - MOS gate delta resistors
H - To detect fuse configurations in the layout

Net: min_lsb[5]

Node1           Node2          Resistance  Layer  Length    Width     Type
=====           =====          ==========  =====  ======    =====     ====
min_lsb/U24/X   min_lsb[5]:24  0.000001     M1    0.000000 10.000000  B
```

---

report_ocvm

Reports information about AOCV or POCV derating tables and coefficients; shows details about path-based and graph-based calculations.

Syntax

```
status report_ocvm
   [-early]
   [-late]
   [-rise]
   [-fall]
   [-clock]
   [-data]
   [-voltage voltage_value]
   [-cell_delay]
   [-net_delay]
   [-lib_cell]
   [-list_annotated]
   [-list_not_annotated]
   [-coefficient]
   -type aocvm | pocvm
   [-nosplit]
   [object_list]
```

Data Types

```
voltage_value     string
object_list       list
```

Feedback

Arguments

`-early`

Shows objects annotated with the AOCV or POCV early deratings.

`-late`

Shows objects annotated with the AOCV or POCV late deratings.

`-rise`

Shows objects annotated with the AOCV or POCV rising deratings.

`-fall`

Shows objects annotated with the AOCV or POCV falling deratings.

`-clock`

Shows objects annotated with the AOCV or POCV clock deratings.

`-data`

Shows objects annotated with the AOCV or POCV data deratings.

`-voltage` *voltage_value*

Shows objects annotated with the AOCV or POCV voltage deratings.

`-cell_delay`

Shows objects annotated with the AOCV or POCV cell deratings.

`-net_delay`

Shows objects annotated with the AOCV or POCV net deratings.

`-lib_cell`

This option can be used with either the -list_annotated or -list_not_annotated options to report the library cell name instead of the instance name.

`-list_annotated`

Lists the leaf cells and global nets that are annotated with AOCV or POCV derating tables.

`-list_not_annotated`

Lists the leaf cells and global nets that are not annotated with AOCV or POCV derating tables.

`-coefficient`

Shows the AOCV or POCV coefficients. You can specify a collection of lib_cell, lib_timing_arc, and cell objects in the *object_list* to display coefficients annotated only on those objects.

Feedback

`-type aocvm | pocvm`

Specifies the OCV type. The allowed values are

- *aocvm* - Shows AOCV information.
- *pocvm* - Shows POCV information.

`-nosplit`

Prevents split lines when columns overflow.

`object_list`

Specifies either timing_path objects for which path-based metrics are to be reported; or timing_arc objects for which graph-based metrics are to be reported; or design objects on which AOCV or POCV derating tables and coefficients have been annotated.

Description

This command displays the user-specified AOCV or POCV information. The type of information displayed by PrimeTime depends on the type of AOCV or POCV information annotated.

See Also

- report_aocvm
- timing_aocvm_enable_analysis
- timing_pocvm_enable_analysis

---

report_odc_power_savings

Generates power saving reports by identifying the Observability Don't Care (ODC) conditions of the registers.

Syntax

string *report_odc_power_savings*

```
[-clock_gate_type      clock_gate_value_type]
[-min_pwr_savings      power_savings]
[-min_reg_width        register_width]
[-csv csv_file]
[-report_nl_names]
[object_list]
```

Feedback

Data Types

clock_gate_value_type	string
power_savings	float
register_width	integer
csv_file	string
object_list	list

Arguments

`-clock_gate_type` *clock_gate_value_type*

Use this option to specify the type of clock which drives the registers. The clock could be gated type, domain type (non-gated) or both. Default value of *-clock_gate_type* is "all" and other supported values are "gated" and "not_gated".

`-min_pwr_savings` *power_savings*

Use this option to report only those registers whose total power saving, after using the ODC power saving technique, is more than or equal to the "power_savings" value. Default value of *-min_pwr_savings* is 0.0.

`-min_reg_width` *register_width*

Use this option to filter the bundled registers so that the bundled registers having width greater than or equal to "register_width" will be considered for power saving estimate. Default value of *-min_reg_width* is 8.

`-csv` *csv_file*

Use this option to specify the path for the csv file for writing the power savings report in a comma seperated value format. A csv file can be viewed in a spreadsheet reader. With this option the report will not be printed on the shell.

`-report_nl_names`

Use this option to report the netlist(gate-level) name of the register. By default, the RTL name or merged RTL name, in case of bus register, is reported for the registers.

object_list

Use this option to apply the ODC power savings techniques on the list of registers. Options *-min_reg_width* and *-min_pwr_savings* cannot be supported with this option. Currently only the gate-level register name can be specified.

Description

Use the *report_odc_power_savings* command to generate the power savings report after applying the ODC power savings techniques on the register/bus registers in the design.

The three important fields that the command prints are the name of registers, potential power savings that could happen using the above technique and the name of Child Registers whose enable values cause this ODC Condition.

The other fields are: width of register bus, frequency of clock, the register is gated or not, the current and final register gate efficiency percentage, current and final enable probability, current power and the final power.

To use the command, you should first apply the update_power and update_metrics command or apply the compute_metrics command.

ODC Power savings Technique

The Observability Don't Care (ODC) technique is one of the sequential clock-gating techniques to find a new enable or strengthen the existing enable for gated registers. When the downstream register is stable (disabled), the data coming to upstream register is not observable at the output. Hence, the upstream registers can be gated to save the clock power. Let us assume that for a given non-gated input register, if children register enable are stable(disabled), then the input register can be gated using the enable of children, avoid the switching activity of its clock and hence saving the power. This could be done at the power overhead of integrated clock gate cell. This command exactly does this and currently it only looks at the ODC condition of the immediate children of the registers/ bus registers and reports them if there is any power savings.

Register Name in report

The command prints the RTL register name or the merged RTL names in case of bus registers by default. If mapping information does not exist for some registers the gate level name would get printed instead.

To print the gate-level register names instead of the RTL names use the option -report_nl_names.

Examples

The following example shows a list of bundled registers having a width >=8, where the power could be saved by employing ODC Techniques. The registers may have either a gated clock or not.

```
pt_shell> report_odc_power_savings -clock_gate_type all
****************************************
Report : report_odc_power_savings
Design : top
Version: U-2022.12
Date   :
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
-----------------------------------
Register                              Register            Gated
   Clock Pin           Current              Current              Current
           Final               Final               Final
 Predicted           Enable
Name                                  Width                (Y/N)
   Frequency           Register Gating     Enable               Power
           Register Gating     Enable              Power
 Clock Power         Expression

   (MHz)               Efficiency (%)      Probability          (W)
           Efficiency (%)      Probability         (W)
 Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------
top/dout/q_out_reg[0:60, 62:63, 78:137, 140:141]
                                      125                 Not Gated
   800.0               0.0                 1                   0.0001416
          100.0               0.0004              7.26e-05
    6.897e-05           FF_EN_-1(top/abc/core/inst_reg[0:60, 62:123,
 126:127]) ||


                 FF_EN_-1(top/abc/core/data_reg[0:31]) ||


                 FF_EN_-1(top/abc/core/data_out[16:20, 22:30]) ||


                  FF_EN_-1(top/abc/core/mem_reg[0][0:60, 62:63, 78:137,
 140:141]) ||


                  FF_EN_-1(top/abc/core/mem_reg[1][0:60, 62:63, 78:137,
 140:141]) ||


                       FF_EN_-1(top/abc/core/inst_r_reg[0:60, 62:123,
 126:127]) ||


                  FF_EN_-1(top/abc/core/mem_reg[2][0:60, 62:63, 78:137,
 140:141]) ||
```

```
                  FF_EN_-1(top/abc/core/data_out[0:5, 7, 9:15]) ||


                   FF_EN_-1(top/abc/core/mem_reg[3][0:60, 62:63, 78:137,
 140:141])

top/abc/core/addr_r_reg[7:11, 15, 18, 20, 22, 26, 31]
                                    11                  Gated
   800.0               70.5                0.3                  1.012e-05
          91.2                  0.09                4.548e-06
 5.577e-06           FF_EN_-1(top/abc/core/addr_r[7:11, 15, 26]


                      top/abc/core/invalid_tag_r_reg


                      top/abc/core/addr_r[18, 20, 22, 31])

top/abc/core/code_r[21, 36:37, 68:71, 88, 112:115]
                                    12                  Gated
   800.0               0.0                 1                    2.367e-05
          99.0                  0.01                1.91e-05
 4.571e-06           FF_EN_-1(top/abc/core/code_r_reg[88]


                      top/abc/core/op_r


                      top/abc/core/code_r[21, 37, 68:71]


                      top/abc/core/inc_r_reg[0:3])

top/xyz/core/target_r_reg[1, 4, 7:8, 10:11, 14:15, 17:21, 23:28, 30:31]
                                    21                  Gated
   800.0               0.0                 1                    3.243e-05
          99.7                  0.003               2.859e-05
   3.84e-06            FF_EN_-1(top/xyz/core/target[1, 4, 7:8, 10:11,
 14:15, 17:21, 23:28, 30:31])

top/abc/core/addr_r_reg[4:5, 12:14, 16:17, 19, 21, 23:25, 27:30]
                                    16                  Gated
   800.0               70.6                0.3                  7.747e-06
```

```
           91.2                 0.09                 3.969e-06
 3.779e-06           FF_EN_-1(top/abc/core/addr_r[4:5, 12, 23:24]


                           top/abc/core/addr_r[13:14, 16:17, 19, 21, 25,
 27:30])

top/xyz/core/addr_r_reg[4:5, 12:20, 22:24, 27:31]
                                  19                   Gated
   800.0               70.1                 0.3                  8.436e-06
           91.0                 0.09                 6.112e-06
  2.323e-06           FF_EN_-1(top/xyz/core/addr_r[4:5, 12:20, 22:24,
 27:31])

top/uvw/m_reg[88:102]
                                  15                   Gated
   800.0               100.0                0                    2.674e-06
           100.0                0                    1.476e-06
 1.198e-06           FF_EN_-1(top/s_reg[35:52]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------------------
```

See Also

- compute_metrics
- update_power
- update_metrics
- report_power
- report_rtl_metrics
- report_est_power_savings
- report_user_def_clock_savings
- report_stc_power_savings

report_parasitics_range

Reports the two factors of both parasitic resistance and capacitance range previously set by set_parasitics_range command.

Syntax

status *report_parasitics_range*

Feedback

Description

The *report_parasitics_range* command reports the two factors of both parasitic resistance and capacitance range previously set by set_parasitics_range.

Examples

The following example reports the parasitic scaling factors.

```
pt_shell> report_parasitics_Range

****************************************
Report : report_parasitics_range

Design : simple1

Version: T-2022.03-SP5-3-DEV-20231025

Date   : Sat Oct 28 10:28:48 2023

****************************************




  Layer list    ground capacitance range    coupling capacitance range
 resistance range      correlation factor


 ------------------------------------------------------------------------
----------------------------------------

  M2            {1.300000 1.000000}         {1.300000 1.000000}
 {0.750000 1.000000}      -1.000000

  M1            {1.200000 1.000000}         {1.200000 1.000000}
 {0.800000 1.000000}      -1.000000

  M3            {1.250000 1.000000}         {1.250000 1.000000}
 {0.850000 1.000000}      -1.000000


 ------------------------------------------------------------------------
----------------------------------------
```

See Also

- read_parasitics
- set_parasitics_range

Feedback

report_path_group

Reports path_group information.

Syntax

string *report_path_group*

```
[-nosplit]
[path_group_list]
```

Data Types

path_group_list	list

Arguments

`-nosplit`

Does not split lines if a column overflows.

path_group_list

Displays the path group information for the *path_group_list* that is specified.

Description

Produces a report showing information about path groups in the *current_design* command. The report includes path groups automatically created by the *create_clock* command and those manually created with the *group_path* command. Path groups are used to affect the calculation of the Maximum Delay cost during optimization. The cost of each group is displayed using the *report_constraint* command.

Examples

The following command displays all the path groups in the current design.

```
pt_shell> report_path_group

****************************************
Report : path_group
Design : test
Version: A-2007.12-DEV
Date   : Sun Jul 29 12:03:16 2007
****************************************

Path_Group     Weight   From      Through        To
------------------------------------------------------------
**default**      1.00   -         -              -
C1               1.00   *         *              C1
C2               1.00   *         *              C2
CLK              1.00   *         *              CLK
```

Feedback

See Also

- create_clock
- current_design
- group_path
- report_constraint
- reset_design

---

report_path_tag_set

Reports timing path tag sets created by the *create_path_tag_set* command.

Syntax

```
status report_path_tag_set
  [-name tag_names]
  [-show_pins]
```

Data Types

```
tag_names  list
```

Arguments

`-name tag_names`

Specifies a list of path tag sets to report. Without this option, the command reports all defined path tag sets.

`-show_pins`

Shows the path endpoint pins of the paths in the reported path tag sets, along with the number of tagged paths that end at each endpoint, listed in order of decreasing number of paths per endpoint.

Description

This command shows a list of the names of the currently defined timing path tag sets, along with the number of paths tagged for each tag name.

Examples

The following example shows that the path tag set named tag1 is associated with 500 timing paths, while the IN1 path tag set is associated with 10 timing paths.

```
pt_shell> report_path_tag_set
****************************************
Report : Path Tag Set
 ...
```

```
****************************************

Tag name                  |  Number of paths
-----------------------------------------
tag1                      |  500
N1                        |  10

1
```

The following example shows the path endpoint pins of the tagged paths along with the number of tagged paths associated with each endpoint.

```
pt_shell> report_path_tag_set -show_pins
 ...
****************************************
Report : Path Signature Count per Endpoint
 ...
****************************************

Number of endpoints: 182

Pin name                     |   Number of paths
-----------------------------------------------
TOP/I_BLDR/s4_reg[14]/D      |  16
TOP/I_BLDR/s4_reg[12]/D      |  16
TOP/I_BLDR/s2_reg[2]/D       |  8
TOP/I_BLDR/s2_reg[2]/D       |  8
pad[0]                       |  1
pad[1]                       |  1
 ...
```

See Also

- create_path_tag_set
- remove_path_tag_set
- enable_path_tagging

---

report_point_to_point_resistance

Reports the point-to-point resistance (in ohms) for all combinations of pins and instance ports for specified nets.

Syntax

status *report_point_to_point_resistance*

```
-of_objects nets
[-limit no_pairs]
```

Feedback

Data Types

```
nets          list
no_pairs      integer
```

Arguments

`-of_objects` *nets*

A list of one or more nets for which to report point-to-point resistance. At least one net is required. Glob-style wildcards (*) are supported.

`-limit` *no_pairs*

Maximum number of resistances to report for each net. The default is 1000.

Description

This command reports the point-to-point resistance (in ohms) for all combinations of pins and instance ports for the specified nets.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a point-to-point resistance report.

```
pt_shell> report_point_to_point_resistance \
            -of_objects min_msb/conv_blk1/n105 -limit 3

************************************************
Report : Point to Point Resistance
Design : toprt
Version: Q-2019.12
Date   : Mon Nov 11 12:45:00 2019
************************************************

Net: min_msb/conv_blk1/n105

Pin1                        Pin2                        P2P R
====                        ====                        =====
min_msb/conv_blk1/U7/X      min_msb/conv_blk1/U10/A     0.025563
min_msb/conv_blk1/U7/X      min_msb/conv_blk1/U11/C     0.035199
min_msb/conv_blk1/U7/X      min_msb/conv_blk1/U14/C     0.024903
```

report_port

Displays port information within the design.

Syntax

string *report_port*

```
[-verbose]
[-design_rule]
[-drive]
[-input_delay]
[-output_delay]
[-wire_load]
[-nosplit]
[-sms_scenarios sms_scenarios_list]
[port_list]
```

Data Types

```
sms_scenarios_list  collection
port_list           list
```

Arguments

`-verbose`

Indicates that the port report includes all port information. By default, only a summary section is displayed that lists all ports and their direction.

`-design_rule`

Reports only port design rule information, including maxCap, manLoad, and maxFanout.

`-drive`

Reports only drive resistance, input transition time, and driving cell information for only input and inout ports.

`-input_delay`

Reports only the port input delay information you set.

`-output_delay`

Reports only the port output delay information you set.

`-wire_load`

Reports only the port wire load information.

`-nosplit`

Prevents line splitting if column overflows. Most design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only port information compatible to the specified SMS scenarios collection are reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

port_list

Displays information on these ports in the current design. Each element in this list is either a collection of ports or a pattern matching the port names.

Description

Displays information about ports in the current design. By default, the command produces a brief report including all ports in the design.

The input and output delay information lists the minimum rise, minimum fall, maximum rise, maximum fall delays, and the clock to which the delay is relative. If "(f)" follows the clock name, the delay is relative to the falling edge of the clock; otherwise, the delay is relative to the rising edge. If "(l)" follows the clock name, input delay is considered to be coming from a level-sensitive latch, or output delay is considered to be going to a level-sensitive latch. By default, the delay is relative to a rising edge-triggered device.

Some information, such as whether the port is in an ideal network or not, is displayed after a timing update (as a result of manually executing an *update_timing* function or experiencing an implicit timing update as a result of executing other commands). This behavior is intended to help you to obtain information and statistics about ports without incurring the cost of a complete timing update.

Examples

This example shows the default port report.

```
pt_shell> report_port
****************************************
Report : port
Design : middle
Version: Y-2006.06
Date   : Wed Mar  8 07:26:43 2006
****************************************

Attributes:
    I - ideal network

                         Pin      Wire
Port           Dir       Cap      Cap     Attributes
-------------------------------------------------------
CLOCK          in      0.0000   0.0000
OPER           in      0.0000   0.0000
in0            in      0.0000   0.0000
in1            in      0.0000   0.0000
```

```
in2             in      0.0000   0.0000
out_1           out     0.0000   0.0000
out_2           out     0.0000   0.0000
out_3           out     0.0000   0.0000
out_4           out     0.0000   0.0000
p_in            in      0.0000   0.0000
p_out           out     0.0000   0.0000

1
```

See Also

- report_design
- report_hierarchy
- report_net
- report_cell
- report_reference

---

report_power

Generates power reports.

Syntax

string *report_power*

```
[-net_power]
[-cell_power]
[-threshold_voltage_group]
[-lvth_groups low_vth_list]
[-leaf]
[-include_boundary_nets]
[-verbose]
[-nworst number]
[-sort_by sort_method]
[-nosplit]
[-hierarchy]
[-levels level]
[-power_greater_than threshold]
[-clocks clock_list]
[-groups group_list]
[-include_estimated_clock_network]
[-derate]
[-rails rails_supply_nets_name]
[-pattern_priority pattern_list]
[-attribute attribute_name_for_cell_pattern]
[-pst]
[-relative_toggle_rate]
```

Feedback

```
[-significant_digits digits]
[-include_shutdown_cells]
[-separate_glitch_power]
[-separate_glitch_power_component]
[-through_switch]
[-include_lib_cells]
[-separate_switching_power]
[-area]
[-exclude_wire_load_info]
[-per_clock per_clock_list]
[-compact_view]
[-unit power_unit]
[object_list]
```

Data Types

low_vth_list	list
number	integer
sort_method	string
level	integer
threshold	float
clock_list	list
group_list	list
rails_supply_nets_name	list
pattern_list	list
attribute_name_for_cell_pattern	string
object_list	list
digits	integer
per_clock_list	list
power_unit	string

Arguments

`-net_power`

Use this option to generate the net-based power report.

`-cell_power`

Use this option to generate the cell-based power report.

`-threshold_voltage_group`

Use this option to report the leakage power per voltage threshold group, instead of reporting the number of cells per voltage threshold group using the *report_threshold_voltage_group* command. You can also use this option in conjunction with the standard filtering options such as *-clocks*, *-groups*, and *-rail*.

`-lvth_groups low_vth_list`

Use this option to specify the list of threshold voltage groups to be considered as low threshold voltage. The value of the argument is a list of threshold voltage group identifiers. Each identifier is a non-empty string that might consist of

alphanumeric characters and the underscore character ("_"). If this option is omitted, no low threshold voltage groups are identified with the "L" attribute. This option is valid only with *-threshold_voltage_group* option.

`-leaf`

Use this option to indicate that the power report must traverse the hierarchy and report the nets or cells at lower-levels (as if the design's hierarchy were flat). The default is to report objects at only the current level of hierarchy. When you use this option in combination with the *-levels* option, the lower levels of the design hierarchy are traversed only to the depth you have specified.

`-include_boundary_nets`

Use this option to indicate that the switching power of primary input nets must be counted when generating the power report; the default is to exclude boundary input nets.

`-verbose`

Use this option to report verbose information, mainly in the header of the report, such as operating conditions, wire load models used to calculate power and the power unit information.

`-nworst` *`number`*

Use this option to filter the report so that it displays only the top-most *number* of the nets or cells sorted by a certain type of sort method.

`-sort_by` *`sort_method`*

Use this option to specify the sorting mode for the net or cell order in the power report.

`-nosplit`

Use this option to prevent line-splitting or section-breaking. By default, if the information for a given field exceeds its fixed column width, the next field begins on a new line, starting in the correct column (line-splitting). By default, if the information for one line exceeds the 80 character limit, the report is broken into two sections, each containing part of the information (section-breaking).

`-hierarchy`

Use this option to generate the hierarchy-based power report.

`-levels` *`level`*

Use this option to specify the number of hierarchy levels to be displayed in the hierarchy-based power report. When you use the *-leaf* option, the number of levels to traverse for the net and cell based report is also specified.

`-power_greater_than` *`threshold`*

Use this option to report only the nets or cells with total power value equal to or greater than *threshold* value.

`-clocks` *`clock_list`*

Use this option to report only the nets, cells, or both nets and cells that belong to the clock domains specified by the *clock_list* value.

`-groups` *`group_list`*

Use this option to report only the nets, cells, or both nets and cells that belong to the power groups specified by the *group_list* value.

`-include_estimated_clock_network`

Use this option to include the clock network power estimated by the *estimate_clock_network_power* command in the power report.

`-derate`

Use this option to include the effective power derate factors in the cell reports. It does not affect other types of reports.

`-rails` *`rails_supply_nets_name`*

Use this option to specify a list of supply nets under UPF mode or design rails under rail mapping mode for which the power values are reported.

`-pattern_priority` *`pattern_list`*

Specifies the list of patterns of cell name (or its attribute value) to be processed for generating the threshold voltage group leakage report.

`-attribute` *`attribute_name_for_cell_pattern`*

Specifies the attribute name, whose values are processed based on the pattern_priority list for generating the threshold voltage group leakage report.

`-pst`

Use this option to report power for valid power state combinations defined at the top level design.

`-relative_toggle_rate`

Use this option with *-net_power* option. When this option is used, the toggle rate specifies the relative toggle rate with respect to the related clock for the given design object.

`-significant_digits` *`digits`*

Specifies the number of digits of precision to be displayed by warnings that show floating point numbers. Allowed values are 0-13; the default is 4.

`-include_shutdown_cells`

Use this option to report shutdown cells as well for valid power state combinations defined at the top level design.

`-separate_glitch_power`

Allows the user to report glitch power components of internal and switching power separately . This option works in time_based mode when the variable "power_enable_clock_cycle_based_glitch" is enabled. When this option is enabled , for each cell or power group 2 additional columns are reported for glitch internal and glitch switching power. The regular internal and switching power columns exclude glitch power components.This feaure enabled, for each net 2 additional columns are reported for glitch rate and glitch switching power.

`-separate_glitch_power_component`

This option works in time_based mode when the variable "power_enable_clock_cycle_based_glitch" is enabled. When this option is enabled , for each cell or power group 2 additional columns are reported for glitch inertial and glitch transport power. The regular internal and switching power columns include glitch power components.

`-through_switch`

Use this option along with *-rail* option. Allows the user to report power associated with the supply_net after getting propagated through the switches.

`-include_lib_cells`

Use this option to report library cell name along with the cell names. -cell or -cell_power option is mandatory to be added to the command to use this option.

`-separate_switching_power`

Allows the user to split switching power into wire power and pin power. Generates net-based wire, pin and switching power report.

`-area`

Use this option to report cell area along with the cell-based or hierarchy-based power report. -cell_power or -hierarchy option is mandatory to be added to the command to use this option.

`-exclude_wire_load_info`

Use this option along with *-verbose* option to exclude wire load models from the verbose report.

`-compact_view`

Use this option for compact view of cell and rail based power report.

`-per_clock` *`per_Iclock_list`*

Use this option for design based report of each of the clock domain specified in the *per_clock_list* value. No any other option should be provided with this for reporting.

`-unit` *`power_unit`*

Use this option to specify the power unit to be used in the power report. Supported units are mW, uW, nW and pW.

`object_list`

Use this option to specify a list of cells, nets, or both to be displayed in the cell-based, net-based, or both cell- and net-based power report.

Description

Use the *report_power* command to generate a power report. It can be used in nondistributed power analysis or distributed power analysis. In the distributed power analysis, the command in the manager process has no options and only generates a merged power report for top design. The following description is for the normal distributed power analysis (nondistributed power analysis or usage in workers of distributed power analysis).

Generates power reports for the design or a specific hierarchy. The specific hierarchy is determined by the *current_instance* command. Before generating the report, power information is updated if necessary.

Four types of power reports can be generated: summary report, cell-based report, net-based report, and hierarchy-based report.

Summary Power Report

When none of the *-cell_power*, *-net_power*, or *-hierarchy* options are specified, a summary power report is generated. The summary power report displays internal, leakage, switching, and total power, as well as peak power, peak time, glitching power, and X transition power, if available, for the current instance, or the current design if the current instance is the top hierarchy of the current design. The summary power report also reports power for each power group. By default, all instances are placed in only one power group; so the summation of the power of the predefined groups equals the total power consumption of the design.

There are seven predefined power groups:

- io_pad: Cells defined as part of the pad_cell group in the library.
- memory: Cells defined as part of the memory group in the library.
- black_box: Cells with no functional description in the library.

- clock_network: Cells in the clock_network excluding io_pad cells.
- register: Latches and flip-flops driven by the clock network excluding io_pads and black_boxes.
- combinational: Nonsequential cells with a functional description.
- sequential: Latches and flip-flops clocked by signals other than those in the clock network.

See the Power Group Power Reporting section.

To report the total leakage breakdowns, set the *power_report_leakage_breakdowns* variable to *true*.

Cell-Based Power Report

When you specify the *-cell_power* option, the cell-based power is reported. The cell-based power report has two sections: Section one displays the cell internal power, cell leakage power, switching power of nets driven by the cell, total power, and percentage of total power compared with the overall power of all cells displayed in the report.

Section two reports the peak power, peak time, glitching power, and X transition power. Section two only shows up when there are peak power, glitching power, X transition power, or both.

When you specify the *-nosplit* option, the two sections are merged together. The peak power for a cell is only available if the waveform is created for this cell. For more information, see the *update_power* man page.

By default, only cells (including hierarchical cells) in current hierarchy are displayed. The power information displayed for a hierarchical cell is the total effect of all leaf cells contained by the hierarchical cell.

If the *power_report_leakage_breakdowns* variable is set to *true*, an additional section of leakage breakdowns is also be printed out.

If the *-derate* option is specified along with the *-cell_power* option, effective power derate factors are included in the power reports. The effective power derate factors are the ratio of derated power to underrated power of a cell. Power components with zero power derate factors are excluded from calculating effective power derate factors. In such a case, a 'z' attribute is added to the corresponding cell.

If you use the *set_power_analysis_options* command with the *-sdpd_tracking* option, the report shows an additional attribute flag "s" for those cells whose state-dependent and path-dependent (SDPD) switching activities are tracked.

If the *-include_lib_cells* option is specified along with the *-cell_power* option, library cell names are reported along with the cell name.

Net-Based Power Report

When you specify the *-net_power* option, the net-based power is reported. The net-based power report displays VDD, total net load, static probability, toggle rate, and net switching power.

By default, only nets in the current hierarchy are displayed. If you use the *-leaf* option, the report would traverse down all lower hierarchies and report all nets found in each hierarchy. When you specify the *object_list* option, only the nets in the object_list are reported. If a net has more than one net segment in different hierarchies, only the net segment in the top most hierarchy is listed in the report.

Hierarchy-Based Power Report

When you specify the *-hierarchy* option, the hierarchy-based power is reported. The report would be in a hierarchical format, with power information on a block-by-block basis. The hierarchy is shown through indentations. The *-levels* option specifies the number of levels of hierarchy to be displayed. Hierarchy levels deeper than the ones specified in this option are not shown in the report. If the *-leaf* option is specified, the leaf cell reference names are to be reported.

For each hierarchical block, the switching, internal, and leakage power are reported, as well as the total power and the percentage of total power compared with the overall power of the top hierarchy. The top hierarchy is determined by current_instance. Hierarchy-based report also has a second section to display peak power, peak time, glitching power, and X transition power, when necessary. Similarly, if the *-nosplit* option is specified, the two sections are merged together. Peak power for the hierarchy is only available if the waveform is created for this hierarchy.

For more information, see the *update_power* man page.

Rail and Supply Nets Based Power Reports

When you specify the *-rails* option, only the power reports associated with the specified rails or supply nets are generated. The *-rails* option accepts a list of name strings. Each name string can contain one or more rail or power supply net names separated by space. One power report is generated for each name string if it contains valid rails or supply nets. If the name string contains "all" keyword, a report of all rails is generated. Multiple power reports can be generated by using the *report_power* command with different combinations of rail names in the list. The *-rails* option works with other options of the *report_power* command to generate cell, hierarchical, or group-based power reports. For time-based power analysis, peak power reports are always for all rails, regardless of the rail specification. Currenlty peak power in consolidated reports is the highest peak among all rails.

Feedback

Negative Power Reporting

When you specify the *-power_greater_than* option along with an appropriate negative threshold value with the *-cell_power* option, the leaf cells with negative power are reported.

Sorting and Filtering

When you specify a certain sort mode with the *-sort_by* option, the list of cell and nets in the cell- based or net-based power report can be sorted. The available sorting modes for the *-net_power* or *-cell_power* option as following:

```
-net_power option          -cell_power option
   -----------------          ------------------
   name                       name
   net_static_probability     cell_internal_power
   net_switching_power        cell_leakage_power
   net_toggle_rate            dynamic_power
   total_net_load             total_power
   total_power
```

If both the *-net_power* and *-cell_power* options are specified and a sorting mode is explicitly selected, the selected sorting mode is used for both the cell and nets reports. Therefore, you must select a sorting mode that applies to both the *-net_power* and *-cell_power* options.

When reporting with the *-hierarchical* option, the same sorting modes are available as the *-cell_power* option. The report is first sorted by hierarchy and then sorted by the requested sorting method, preserving the hierarchical structure of the report.

If the sorting mode is not explicitly set, a default is chosen based on the mode of the *report_power* command:

```
Mode                          Implicit default
      ----------                    ------------------
      -net_power                    net_switching_power
      -cell_power                   cell_internal_power
      -net_power -cell_power        dynamic_power
```

The sorted list of the cells and nets in the cell-based and net-based power report can be filtered to display only the top most number of cells or nets specified by the *-nworst* option.

Clock Domain Power Reporting

When you specify the *clock_list* value with the *-clocks* option, a power report is generated for the cells and nets in the specified clock domains.

The cells and nets in the clock tree network of CLK clock are taken as belonging to CLK clock domain. The cells and nets in a timing path launched by the CLK clock are regarded as belonging to CLK clock domain. The cells and nets in a timing path with no launch clock, but captured by the CLK clock are regarded as belonging to the CLK clock domain

as well. If a cell or net belongs to multiple clock domains, it is forced to belong to the clock domain with the fastest clock. If a cell or net belongs to no clock domains, it is forced to belong to the fastest clock domain in the design.

The power for the design or a hierarchy is no longer the total effect of all leaf cells in the design or hierarchy, but instead, the total effect of all leaf cells that are in the design or hierarchy AND belong to the specified clock domains.

Clock domain power reporting supports all of the four types of power report.

Power Group Power Reporting

When you specify the *group_list* value with the *-groups* option, a power report is generated for the cells and nets in the specified clock domains.

Power groups can be created by the *create_power_group* command. The tool has also several predefined power groups. For more information, see the *create_power_group* man page.

The power for the design or a hierarchy is no longer the total effect of all leaf cells in the design or hierarchy, but instead, the total effect of all leaf cells that are in the design or hierarchy AND belong to the specified power groups.

Power group power reporting supports all of the four types of power report. In the summary power report, if the *-groups* option is not specified, all the power groups are reported. For each power group, the cumulative switching, internal, and leakage power of the power group are reported, as well as the total power and the percentage of the total power compared with the overall power of the top hierarchy. The top hierarchy is determined by current_instance. This power group report also has a second section to display the peak power, peak time, glitching power, and X transition power, when necessary.

Similarly, if the *-nosplit* option is specified, the two sections are merged together. Peak power for the power group is only available if the waveform is created for this power group. For more information, see the *update_power* man page.Option *-nosplit* is supported along with option *-significant_digits*.

Clock Tree Power Reporting

Clock tree power and register (driven by the clock tree) power can be reported using the *report_power* command by using the power group power reporting. There are two predefined power groups: clock_network and register. The clock_network power group contains all cells in the clock tree network. The register power group contains all registers driven by the clock tree network.

The *power_clock_network_include_clock_gating_network* and *power_clock_network_include_register_clock_pin_power* variables can affect clock tree power reporting. For more information, see the man pages.

The clock tree power reporting supports all of the four types of power report. By default, the summary power report displays power of the predefined clock_network and register power groups. An attribute label "i" is used to indicate whether the clock_network or register power group includes the register clock pin power. In the cell-based power report, an attribute label "c" is used to indicate that only clock pin internal power is displayed for the register cell; a "d" attribute label is used to indicate that the power displayed for the register cell does not include the register clock pin internal power.

The clock tree power for certain clocks (if not all clocks) can be reported by combining the *-groups* and *-clocks* options.

Estimated Clock Tree Power Reporting

Clock tree power and register (driven by the clock tree) power estimated using the *estimate_clock_network_power* command can be reported using the *-include_estimated_clock_network* option. Estimated clock tree power can only be reported in the Summary Report; therefore, the *-cell_power* and *-net_power* options cannot be specified together with the *-include_estimated_clock_network* option.

The power of the existing clock tree network, if any, is subtracted from the total power, and the power of the estimated clock tree power is added. If the estimated clock tree power includes register power by using the *estimate_clock_network_power -include_registers* command, the power of the existing registers is subtracted from the total power and the power of the estimated register power is added. Accordingly, the power for the power groups "clock_network" and "register" should be affected. These power groups do not include estimated clock network power. However, the power for the existing clock network is subtracted.

A new section in the Summary power report section shows the estimated clock network power for each clock that has been estimated. The *estimate_clock_network_power* command must be used before the *report_power* command. Multiple *estimate_clock_network_power* commands can be used, but only the power from the latest command that estimates clock tree power for a certain clock is reported by the *report_power* command.

The *-clocks* option can be specified together with *-include_estimated_clock_network* option. In this case, only the clock tree power estimated for the specified clocks is included in the power report.

The *-groups* option can be specified together with *-include_estimated_clock_network* option. In this case, only when the *clock_network* and *register* power groups are specified, the estimated clock network power is reported.

Low-Level Hierarchy Power Reporting

Power reports can be generated for the top design and a lower-level hierarchy. The hierarchy can be determined by *current_instance*. The power reports for the top design are special cases, when the current instance is the top-level hierarchy of the design. If the

current instance is not the top design, only the cells or nets under the current instance are targeted for power reporting.

To generate different power reports for each clock domain, power group, and hierarchy, combine the *-clocks* and *-groups* options with using the *current_instance* command.

PST based Power Reports

When you specify the *-pst* option, PST based power analysis reports are generated. It reports power for valid power state combinatios defined at the top level design. This feature requires that the variable *upf_enable_pst* is set to *true* before load_upf. Option *-pst* can be specified with other *report_power* options to generate cell and hierarchical power reports. But it cann't be specified with *-rails* option. These two reportng features are mutually exclusive.

Examples

The following example shows a verbose summary power report.

```
pt_shell> report_power -verbose
****************************************
Report : power
        -verbose
 ...
****************************************
Library(s) Used: power_lib (File: /remote/libraries/power_lib.db)

Operating Conditions:
Wire Loading Model Mode: enclosed

Cell    Design           Wire Loading Model      Library
------------------------------------------------------------
        mydesign            0.5K_TLM          power_lib.db
sub     submodule           0.5K_TLM          power_lib.db
------------------------------------------------------------

Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W

  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                 Internal  Switching  Leakage    Total
Power Group      Power     Power      Power      Power     (     %)  Attr
-----------------------------------------------------------------------
---
```

```
io_pad             0.0000    0.0000     0.0000     0.0000    ( 0.00%)
memory             0.0000    0.0000     0.0000     0.0000    ( 0.00%)
black_box          0.0000    0.0000     0.0000     0.0000    ( 0.00%)
clock_network   1.813e-04  4.199e-05  4.192e-10  2.233e-04 (70.03%)  i
register        8.442e-05  1.114e-05  9.208e-09  9.557e-05 (29.97%)
combinational      0.0000    0.0000     0.0000     0.0000    ( 0.00%)
sequential         0.0000    0.0000     0.0000     0.0000    ( 0.00%)
--------------------------------------------------------------------------------
---

  Net Switching Power  = 5.313e-05   (16.66%)
  Cell Internal Power  = 2.657e-04   (83.33%)
  Cell Leakage Power   = 9.627e-09   ( 0.00%)
                         ---------
Total Power            = 3.188e-04  (100.00%)

1
```

The following example shows a net-based power report sorted by the *net_switching_power* value and filtered to display only the five nets with highest switching power.

```
pt_shell> report_power -net_power -leaf -nworst 5
****************************************
Report : power
        -net_power
        -nworst 5
        -leaf
        -sort_by net_switching_power
 ...
****************************************


  Attributes
  ----------
      a  -  Switching activity information annotated on net
      p  -  Propagated switching activity information on net
      d  -  Default switching activity used on net
      u  -  Net switching activity uninitialized
      m  -  Net is driven by multiple pins

                       Total      Static   Toggle   Switching
Net              Vdd   Net Load   Prob.    Rate     Power      Attrs
--------------------------------------------------------------------
net36            1.80    0.026    0.248    0.1985   8.397e-06  p
net42            1.80    0.026    0.248    0.1985   8.397e-06  p
net48            1.80    0.026    0.248    0.1985   8.397e-06  p
net54            1.80    0.026    0.248    0.1985   8.397e-06  p
sub/net20        1.80    0.026    0.248    0.1985   8.397e-06  p
--------------------------------------------------------------------
Total (5 nets)                                      4.199e-05 Watt
```

```
1
```

The following example shows a cell-based power report for clock_network power group (clock tree power).

```
pt_shell> report_power -cell_power -groups clock_network
****************************************
Report : power
        -cell_power
        -sort_by cell_internal_power
        -groups clock_network
Design : mydesign
 ...
****************************************
  Attributes
  ----------
      a  -  Annotated internal & leakage power
      b  -  Black box (unresolved) cell
      c  -  Clock pin internal power only
      d  -  Does not include clock pin internal power
      h  -  Hierarchical cell

              Internal  Switching  Leakage    Total
Cell          Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------------------
sub           3.626e-05 8.397e-06 8.384e-11 4.466e-05 (20.00%)  h
clk_out1_reg  1.228e-05 8.397e-06 8.384e-11 2.068e-05 ( 9.26%)  h
clk_temp1_reg 1.228e-05 8.397e-06 8.384e-11 2.068e-05 ( 9.26%)  h
temp1_reg_3_  5.995e-06    0.0000    0.0000 5.995e-06 ( 2.68%)  c
temp1_reg_0_  5.995e-06    0.0000    0.0000 5.995e-06 ( 2.68%)  c
--------------------------------------------------------------------------------
Totals        1.813e-04 4.199e-05 4.192e-10 2.233e-04 (100.0%)
--------------------------------------------------------------------------------
1
```

The following example shows a cell-based power report where library cell name is reported along with the cell name using -include_lib_cells option.

```
pt_shell> report_power -cell_power -include_lib_cells
****************************************
Report : Averaged Power
    -cell_power
    -include_lib_cells
    -sort_by cell_internal_power
Design : mac
 ...
****************************************

Attributes
----------
      a  -  Annotated internal | leakage | switching power
      b  -  Black-box (unresolved) cell
```

```
      c  -  Clock pin internal power only
      d  -  Does not include clock pin internal power
      h  -  Hierarchical cell

                        Lib Cell                        Internal  Switching
  Leakage    Total
Cell                    Name                            Power     Power
  Power      Power   (     %)  Attrs
--------------------------------------------------------------------------
-------------------------------------
mult_21                 mac_DW02_mult_16_16_0           2.655e-04 3.171e-04
 1.491e-07 5.828e-04 (70.03%)  h
add_23                  mac_DW01_add_33_0               1.005e-04 1.489e-04
 2.362e-08 2.494e-04 (29.97%)  h
--------------------------------------------------------------------------
-------------------------------------
Totals (2 cells)                                        3.660e-04 4.660e-04
 1.727e-07 8.322e-04 (100.0%)
1
```

The following example shows a cell-based power report where cell area is reported using -area option.

```
pt_shell> report_power -cell_power -area
****************************************
Report : Averaged Power
 -cell_power
 -sort_by cell_internal_power
 -area
Design : top
 ...
****************************************



  Attributes
  ----------
      a  -  Annotated internal | leakage | switching power
      b  -  Black-box (unresolved) cell
      c  -  Clock pin internal power only
      d  -  Does not include clock pin internal power
      h  -  Hierarchical cell

                        Internal  Switching Leakage   Total
Cell                    Power     Power     Power     Power    (     %)
    Area   Attrs
--------------------------------------------------------------------------
-----------------
u1                      8.032e-07 1.134e-05 3.302e-07 1.248e-05 (98.58%)
    3.2040 h
u0                      1.041e-07 7.590e-08 2.130e-10 1.802e-07 ( 1.42%)
    1.4400
```

```
------------------------------------------------------------------------
-----------------
Totals (2 cells)        9.073e-07 1.142e-05 3.304e-07 1.266e-05 (100.0%)
    4.6440
1
```

The following example shows a hierarchy-based power report for the "sub" instance.

```
pt_shell> current_instance sub
pt_shell> report_power -hierarchy -levels 2
****************************************
Report : power
        -hierarchy
        -levels 2
Design : mydesign/sub (submodule)
 ...
****************************************

                            Switch   Int      Leak      Total
Hierarchy                   Power    Power    Power     Power    %
------------------------------------------------------------------------
sub (submodule)             9.98e-06 5.33e-05 1.93e-09  6.33e-05 100.0
  clk_gate_out_reg (SNPS_CLOCK_GATE_HIGH_submodule)
                            9.14e-06 1.64e-05 2.41e-10  2.56e-05  40.4
------------------------------------------------------------------------
1
```

The following example shows a report including estimated clock-network power.

```
pt_shell> report_power -include_estimated_clock_network
****************************************
Report : power
 ...
****************************************
  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                         Internal  Switching  Leakage    Total
Power Group              Power     Power      Power      Power     (%)
 Attrs
------------------------------------------------------------------------
-------
io_pad                   0.0000    0.0000     0.0000     0.0000
  (0.00%)
memory                   0.0000    0.0000     0.0000     0.0000
  (0.00%)
black_box                0.0000    0.0000     0.0000     0.0000
  (0.00%)
clock_network            0.0000    0.0000     0.0000     0.0000
  (0.00%)
```

```
register                 8.442e-05 1.114e-05  9.208e-09  9.557e-05
 (29.97%)   i
combinational            0.0000    0.0000     0.0000     0.0000
 (0.00%)
sequential               0.0000    0.0000     0.0000     0.0000
 (0.00%)
--------------------------------------------------------------------------
---------
  Attributes
  ----------
      i  -  Including driven register power
                         Internal  Switching  Leakage    Total
Clock                    Power     Power      Power      Power    (%)
 Attrs
--------------------------------------------------------------------------
-------
clk                      1.813e-04 4.199e-05  4.192e-10  2.233e-04
--------------------------------------------------------------------------
-------
Estimated Clock          1.813e-04 4.199e-05  4.192e-10  2.233e-04(70.03%)
--------------------------------------------------------------------------
---------

  Net Switching Power  = 5.313e-05  (16.66%)
  Cell Internal Power  = 2.657e-04  (83.33%)
  Cell Leakage Power   = 9.627e-09  ( 0.00%)
-------------------------------------------------
Total Power            = 3.188e-04  (100.00%)
-------------------------------------------------
1
```

The following example shows the leakage variation report.

```
pt_shell> report_power
****************************************
Report : Leakage Variation Report
Design : test_design
 ...
****************************************
Library(s) Used: power_variations_lib
 (File: /remote/libraries/power_variations_lib.db)

Operating Conditions:
Wire Load Model Mode: unknown
(no wire load model is set)

Power-specific unit information :
    Voltage Units = 1 V
    Capacitance Units = 1 pf
    Time Units = 1 ns
    Dynamic Power Units = 1 W
    Leakage Power Units = 1 W
```

```
------------------------------------------------------------------------
Design                quantile    sensitivity    mean        stddev
------------------------------------------------------------------------
test_design           1.21771e-06 0.416507    6.25407e-07 2.60486e-07
------------------------------------------------------------------------
1
```

The following example shows power reports for rail VDD1 and VDD2, VDD2 and VDD3 and for all rails.

```
pt_shell> report_power -rails {"VDD1 VDD2" "VDD2 VDD3" "all"}
****************************************
Report : Averaged Power
 ...
****************************************
Current Power Rail: all

Power Report For Rails: VDD1 VDD2

  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                    Internal   Switching  Leakage    Total
Power Group         Power      Power      Power      Power      (     %)
 Attrs
------------------------------------------------------------------------
----
io_pad              0.0000     0.0000     0.0000     0.0000     ( 0.00%)
memory              0.0000     0.0000     0.0000     0.0000     ( 0.00%)
black_box           0.0000     0.0000     0.0000     0.0000     ( 0.00%)
clock_network       0.0000     0.0000     0.0000     0.0000     ( 0.00%)
 i
register            0.0000     0.0000     0.0000     0.0000     ( 0.00%)
combinational       4.061e-06  1.800e-07  8.415e-11  4.241e-06 (100.00%)
sequential          0.0000     0.0000     0.0000     0.0000     ( 0.00%)
------------------------------------------------------------------------
-----

  Net Switching Power  = 1.800e-07   ( 4.24%)
  Cell Internal Power  = 4.061e-06   (95.75%)
  Cell Leakage Power   = 8.415e-11   ( 0.00%)
------------------------------------------------------------
Total Power            = 4.241e-06  (100.00%)
------------------------------------------------------------
****************************************
Report : Averaged Power
 ...
****************************************

Current Power Rail: all
```

```
Power Report For Rails: VDD2 VDD3
  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                        Internal   Switching  Leakage    Total
Power Group             Power      Power      Power      Power      (
  %)  Attrs
-------------------------------------------------------------------------
-----------
io_pad                  0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
memory                  0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
black_box               0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
clock_network           0.0000     0.0000     0.0000     0.0000
 ( 0.00%)  i
register                0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
combinational           1.973e-06  1.714e-07  4.356e-11  2.145e-06
 (100.00%)
sequential              0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
-------------------------------------------------------------------------
-----------

  Net Switching Power  = 1.714e-07   ( 7.99%)
  Cell Internal Power  = 1.973e-06   (92.00%)
  Cell Leakage Power   = 4.356e-11   ( 0.00%)
  ------------------------------------------
Total Power            = 2.145e-06  (100.00%)
-------------------------------------------------------------------------
-----------
****************************************
Report : Averaged Power
 ...
****************************************
Current Power Rail: all

  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User-defined power group

                        Internal   Switching  Leakage    Total
Power Group             Power      Power      Power      Power      (
  %)  Attrs
-------------------------------------------------------------------------
----------
io_pad                  0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
```

```
memory                  0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
black_box               0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
clock_network           0.0000     0.0000     0.0000     0.0000
 ( 0.00%)  i
register                0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
combinational           1.842e-05  4.886e-07  9.315e-11  1.891e-05
 (100.00%)
sequential              0.0000     0.0000     0.0000     0.0000
 ( 0.00%)
--------------------------------------------------------------------------------
----------
  Net Switching Power  = 4.886e-07   ( 2.58%)
  Cell Internal Power  = 1.842e-05   (97.42%)
  Cell Leakage Power   = 9.315e-11   ( 0.00%)
 ---------------------------------------------------
Total Power            = 1.891e-05  (100.00%)
--------------------------------------------------------------------------------
----------

1
```

The following example shows a threshold voltage group *report_power* report.

```
pt_shell> report_power -threshold_voltage_group -lvth_groups {LVT}
****************************************
Report : Threshold Voltage Group based leakage power
        -threshold_voltage_group
        -lvth_groups LVT XYZK
 ...
****************************************

Attributes :
------------
         u -> user-defined power group
         L -> user-defined low vth group

Power Group     HVT                 LVT(L)              SVT
   Total
Name            leakage (%)         leakage (%)         leakage (%)
   leakage (%)     Attrs
--------------------------------------------------------------------------------
--------------------------
memory          0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
   0.0000 (  0.00%)
io_pad          0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
   0.0000 (  0.00%)
clock_network   0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
   0.0000 (  0.00%)
black_box       0.0000 (  0.00%)    0.0000 (  0.00%)    0.0000 (  0.00%)
   0.0000 (  0.00%)
```

```
register        5.259e-07 ( 18.86%) 2.263e-06 ( 81.14%)    0.0000 (
  0.00%) 2.789e-06 (100.00%)
combinational   0.0000 (  0.00%) 3.223e-06 ( 87.92%) 4.428e-07 ( 12.08%)
 3.666e-06 (100.00%)
sequential      0.0000 (  0.00%)    0.0000 (  0.00%) 2.175e-07 (100.00%)
 2.175e-07 (100.00%)
--------------------------------------------------------------------------
-------------------------
Total           5.259e-07 (  7.88%) 5.486e-06 ( 82.22%) 6.603e-07 (
  9.90%) 6.672e-06 (100.00%)
--------------------------------------------------------------------------
------------------------
SUMMARY :
        Low Vth leakage (%)       = 5.486e-06 ( 82.22%)
        Other Vth leakage (%)     = 1.186e-06 ( 17.78%)
        Undefined Vth leakage (%) =    0.0000 (  0.00%)
        ------------------------------------------
        Total leakage (%)         = 6.672e-06 (100.00%)
--------------------------------------------------------------------------
--------------------

1
```

The following example shows a separate switching power *report_power* report.

```
pt_shell>  report_power -separate_switching_power {n245 n246}
****************************************
Report : Averaged Power
        -separate_switching_power
 ...
****************************************


                                  Wire       Pin          Total
Net                               Power      Power      Switching Power
--------------------------------------------------------------------------
-------
n245                              4.374e-08  4.602e-08  8.976e-08
n246                              4.312e-10  8.102e-10  1.241e-09
--------------------------------------------------------------------------
-------
Total (2 nets)                                          9.100e-08 Watt

1

pt_shell>  report_power -net_power   [get_nets -of U19]
 -separate_glitch_power
****************************************
Report : Time Based Power
        -net_power
        -sort_by net_switching_power
****************************************
```

```
  Attributes
  ----------
      a  -  Switching activity information / Switching power information
 annotated on net
      p  -  Propagated switching activity information on net
      d  -  Default switching activity used on net
      u  -  Net switching activity uninitialized
      m  -  Net is driven by multiple pins
      b  -  Net is a primary input (boundary net)

                Toggle rates reported with respect to time unit 1ns
------------------------------------------------------------------------------
---------------------------
                                      Total      Static   Toggle   Switching
 Glitch   Glitch
Net                           Vdd   Net Load   Prob.    Rate     Power
 Rate     Power      Attrs
------------------------------------------------------------------------------
---------------------------
c_r[16]                       1.80    0.015    0.505    0.0493   1.210e-06
 0.0000      0.0000  a
m3[16]                        1.80    0.015    0.503    0.0391   9.602e-07
 0.0102   2.499e-07  a
c_rr[16]                      1.80    0.005    0.502    0.0493   4.100e-07
 0.0000      0.0000  a
n245                          1.80    0.602    0.300    1.00e-4     0.0000
 0.0000      0.0000  a
------------------------------------------------------------------------------
---------------------------
Total (4 nets)                                                    2.580e-06
 Watt      2.499e-07 Watt

1

pt_shell>  report_power -rails {VDD} -cell_power {divflop delay2}
 -compact_view
****************************************
Report : Averaged Power
        -cell_power
        -compact_view
        -sort_by cell_internal_power
****************************************


  Attributes
  ----------
      a  -  Annotated internal & leakage power
      b  -  Black-box (unresolved) cell
      c  -  Clock pin internal power only
```

```
      d  -  Does not include clock pin internal power
      h  -  Hierarchical cell

Voltage Supplies:VDD:1.3V

Internal  Switching  Leakage    Total
(VDD)     (VDD)      (VDD)
Power     Power      Power      Power     (     %)     Cell       Attrs
--------------------------------------------------------------------------
----
1.582e-06 1.099e-07 9.000e-05   9.169e-05 (99.45%)     divflop
2.782e-07 1.096e-07 1.163e-07   5.041e-07 ( 0.55%)     delay2
--------------------------------------------------------------------------
----
1.860e-06 2.195e-07 9.012e-05   9.220e-05 (100.00%)

Total(VDD)(2 cells): 9.22e-05 (100.00%)
Total(2 cells): 9.22e-05 (100.00%)
1

pt_shell>  report_power -nosplit -significant_digits 5
****************************************
Report : Time Based Power
  -significant_digits
  -nosplit
****************************************



  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group

                        Internal             Switching
 Leakage               Total                     Peak       Peak
 Glitch              X-tran
Power Group             Power                  Power                  Power
               Power          (     %)   Power      Time          Power
             Power              Attrs
--------------------------------------------------------------------------
--------------------------------------------------------------------------
---------------------------------------
clock_network           6.27721e-04         0.00000e+00
 0.00000e+00         6.27721e-04         (18.48%)     N/A         N/A
   0.00000e+00         0.00000e+00         i
register                4.38157e-04         8.98749e-05
 6.66027e-08         5.28098e-04         (15.55%)     N/A         N/A
   4.79659e-05         3.02019e-06
combinational           8.78444e-04         1.11784e-03
 1.83637e-07         1.99647e-03         (58.79%)     N/A         N/A
   6.64955e-04         4.44105e-06
```

```
sequential              5.14281e-05          1.92350e-04
 9.16323e-09         2.43787e-04          ( 7.18%)     N/A          N/A
   2.07662e-07         0.00000e+00
memory                  0.00000e+00          0.00000e+00
 0.00000e+00         0.00000e+00          ( 0.00%)     N/A          N/A
   0.00000e+00         0.00000e+00
io_pad                  0.00000e+00          0.00000e+00
 0.00000e+00         0.00000e+00          ( 0.00%)     N/A          N/A
   0.00000e+00         0.00000e+00
black_box               0.00000e+00          0.00000e+00
 0.00000e+00         0.00000e+00          ( 0.00%)     N/A          N/A
   0.00000e+00         0.00000e+00

  Net Switching Power  = 1.40006e-03           (41.23%)
  Cell Internal Power  = 1.99575e-03           (58.77%)
  Cell Leakage Power   = 2.59403e-07           ( 0.01%)
                         -------------------------
Total Power            = 3.39607e-03          (100.00%)

X Transition Power     = 7.46124e-06
Glitching Power        = 7.13129e-04

Peak Power             = 4.15677e-01
Peak Time              =    180.00

1

pt_shell>  report_power -unit mW
****************************************
Report : Averaged Power
 -unit mW
Design : TOP
****************************************



  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group

                      Internal  Switching  Leakage    Total
Power Group           Power     Power      Power      Power   (     %)
 Attrs
--------------------------------------------------------------------------------
-------
clock_network         3.469e-03 1.041e-03 1.851e-04 4.696e-03 ( 1.49%)
 i
register              3.818e-02 1.052e-03 4.133e-02 8.057e-02 (25.59%)
combinational         7.573e-02 2.916e-02 1.247e-01 2.296e-01 (72.92%)
sequential               0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                   0.0000    0.0000    0.0000    0.0000 ( 0.00%)
io_pad                   0.0000    0.0000    0.0000    0.0000 ( 0.00%)
```

```
black_box                    0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 3.126e-02   ( 9.93%)
  Cell Internal Power  = 1.174e-01   (37.28%)
  Cell Leakage Power   = 1.662e-01   (52.79%)
                         ---------
Total Power            = 3.149e-01  (100.00%)

1
```

See Also

- create_power_group
- current_instance
- estimate_clock_network_power
- report_power
- set_power_analysis_options
- update_power
- create_power_state_group
- upf_enable_pst
- power_clock_network_include_clock_gating_network
- power_clock_network_include_register_clock_pin_power
- power_report_leakage_breakdowns

---

report_power_analysis_options

Reports the options for power analysis.

Syntax

status *report_power_analysis_options*

Arguments

None.

Description

The *report_power_analysis_options* command reports the default and user settings for power analysis.

The *collection_result_display_limit* variable specifies the number of cells displayed by the *set_power_analysis_options -cell* command.

Examples

The following is an example from the *report_power_analysis_options* command output:

```
pt_shell> report_power_analysis_options

****************************************
Report : Power Analysis Options
 ..
****************************************

Power analysis mode : time_based

  Option Name              Value
-------------------------------------
  include                  top
  waveform_format          fsdb
  waveform_output          ptpx
```

See Also

- report_power
- set_power_analysis_options
- update_power
- collection_result_display_limit
- power_analysis_mode

---

report_power_budget

Reports power budget and derived scaling factors for the hierarchical instances in the design.

Syntax

```
int report_power_budget
```

Description

This command is used to report the power budget and derived scaling factor for the hierarchical instances in the design.

Examples

In the following example, power budget have been set on the following hierarchies.

Feedback

```
pt_shell> set_power_budget 8.513e-05 -cell four_mini_inst/two_mini_1/a
pt_shell> set_power_budget 2.338e-04 -cell four_mini_inst/two_mini_1
 -rails four_mini_inst/VDD_MINI_1
pt_shell> set_power_budget 9.354e-05 -cell four_mini_inst/two_mini_1/a
 -rails four_mini_inst/VDD_MINI_1
pt_shell> report_power_budget
****************************************
Report : power budget
Design : top
****************************************

    Cell Name                     Rail                            Power
 Budget        Scale Factor
 ---------------------------------------------------------------------------
------------------
    four_mini_inst/two_mini_1     four_mini_inst/VDD_MINI_1
 2.104405e-04        2.50
    four_mini_inst/two_mini_1/a   --
 8.513000e-05        1.50
    four_mini_inst/two_mini_1/a   four_mini_inst/VDD_MINI_1
 9.354000e-05        2.00
1
```

See Also

- set_power_budget

---

report_power_calculation

Displays the actual calculation of internal power for a pin, leakage power for a cell, or switching power for a net.

Syntax

integer *report_power_calculation*

```
[-state_condition state]
[-path_sources name_list]
[-rise]
[-fall]
[-verbose]
[-report_output prefix]
[-report_type type]
[-time]
[-sensitivity]
[-exclude_wire_load_info]
object_list
```

Feedback

Data Types

state	string
name_list	string
prefix	string
type	string
object_list	list

Arguments

`-state_condition` *state*

Specifies that the report should be restricted to tables or polynomials with matching state condition. This option can only be specified for objects of type pin or cell. Use the *-state_condition default* option to specify the default state condition. Use the *-state_condition all* option to specify that all states must be considered for reporting.

`-path_sources` *name_list*

Specifies the related input pin that causes the output pin transition. This option can only be specified for objects of type pin. The path information is used not only for picking the internal power table or polynomial, but also for choosing the input transition time for power calculation if the input transition time is one of the variables. Use the *-path_sources default* option to specify the default path source (without related pin). Use the *-path_sources all* option to specify all possible paths.

`-rise`

Specifies that the internal power report should be limited to rise transition only. If not specified, both rise and fall transitions are reported, which is equivalent to specifying both the *-rise* and *-fall* options.

`-fall`

Specifies that the internal power report should be limited to fall transition only. If not specified, both rise and fall transitions are reported, which is equivalent to specifying both the *-rise* and *-fall* options.

`-verbose`

Specify this option to increase the amount of information that is reported for cells or pins. For cells, the leakage power calculation report is appended by an internal power calculation report for all pins and all possible states and path sources of the cells in the object list. For pins, all the possible states and paths are printed, which is equivalent to the *-state_condition all -path_sources all* option.

`-report_output` *`prefix`*

This option is applicable only in time_based mode and it is specifies the prefix of the reports to be generated. By default the report names are ptpx_detailed.rpt and ptpx_summary.rpt .

`-report_type` *`type`*

This option is applicable only in time_based mode and the valid values are 'detailed' , 'summary' and 'both'. If nothing is specified both detailed and summary reports are generated.

`-time`

Specify the time window for which the report is to be generated in time_based mode. If nothing is specified report is generated for the entire simulation time .

`object_list`

Specifies the objects for which the power calculation is reported. The type of power calculation depends on the object type. For a pin, internal power calculation is reported. For a cell, a leakage power calculation report is produced. For a net, a switching power calculation report is produced. In time_based mode , only cell object type is allowed.

`-sensitivity`

Specify this option to control LLE related sensitivity warnings.

`-exclude_wire_load_info`

This option excludes wire load information reported. Used along with *verbose* option.

Description

The *report_power_calculation* command provides detailed power calculation information for the specified pin, cell, or net. You can use this information for debugging or verifying power data in a technology library. Before using this command to display details of internal or leakage power calculation, read the technology library file into the system with the library features attribute, the *report_power_calculation* attribute included in the library - "library_features (report_power_calculation)", which enables the report_power_calculation command for internal, leakage, and net switch- ing power. If the library is missing this feature, then users do not have access to the power table values in library used to calculate power. Therefore, the built-in security mechanism terminates the command. And, details of power calculation information will not be displayed.

In time_based mode this command generates a detailed report on the sequence of events processed for power calculation for the specified list of cells and time window. It also generates a summary report containing the sdpd activites for the specified cells. The

options 'state_condition' , 'rise' , 'fall' and 'path_sources' are not applicable in time_based mode.

PrimePower supports power net based power report feature for multivoltage designs. You can use the *set_current_power_net* command to set the power net of interest. Only the power dissipated on the specified power nets is calculated and reported. By default (unless otherwise specified), all the power nets are included in the power analysis. Use the *get_current_power_net* command to show the current power net setting.

Examples

This example shows the calculation of path dependent internal power from input pin B to output pin Y. The contributions for rise and fall internal power are shown separately.

```
pt_shell> report_power_calculation [get_pin U4/Y] -state default -path B

****************************************
Report : power_calculation
        U4/Y
        -state_condition default
        -path_sources B
Design : rpc
Version: W-2004.12
Date   : Thu Sep  2 12:10:44 2004
****************************************


Library(s) Used:

    example (File: /root/libraries/example.db)


Operating Conditions: typical   Library: example
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
rpc                    a                 example


Global Operating Voltage = 0.9
Library Power-specific unit information :
    Voltage Unit : 1 V
    Capacitance Unit : 1 pF
    Time Unit : 1 ns
    Dynamic Power Unit : 0.001 W    (derived from V,C,T units)
    Leakage Power Unit : 1e-09 W

Note: Unless specified, the numbers reported are in library units.

================================================================================
=======
```

```
Pin Internal Power Calculation
    cell: U4
    pin: Y
    path source: B

  Path Dependent Rise Pin Internal Power = 2.35214e-05 W
    pin internal power = internal energy * pin toggle rate

  Rise internal energy per transition = 0.143423
    library model: NLPM
    table variables:
      X = total_output_net_capacitance = 0.4
      Y = input_net_transition = 0.3
    relevant portion of lookup table:
                    (X)  0.1050      (X)  0.3550
    (Y)  0.0500     (Z)  0.1310      (Z)  0.1410
    (Y)  0.4510     (Z)  0.1320      (Z)  0.1420

       Z = A + B*X + C*Y + D*X*Y
       A =  0.1267          B =  0.0400
       C =  0.0025          D =  0.0000

    Z = 0.143423
    (no scaling since the library has no internal power k-factors)

  Path Dependent Rise Pin Toggle Rate = 0.164

  Path Dependent Fall Pin Internal Power = 2.35214e-05 W
    pin internal power = internal energy * pin toggle rate

  Fall internal energy per transition = 0.143423
    library model: NLPM
    table variables:
      X = total_output_net_capacitance = 0.4
      Y = input_net_transition = 0.3
    relevant portion of lookup table:
                    (X)  0.1050      (X)  0.3550
    (Y)  0.0500     (Z)  0.1310      (Z)  0.1410
    (Y)  0.4510     (Z)  0.1320      (Z)  0.1420

       Z = A + B*X + C*Y + D*X*Y
       A =  0.1267          B =  0.0400
       C =  0.0025          D =  0.0000

    Z = 0.143423
    (no scaling since the library has no internal power k-factors)

  Path Dependent Fall Pin Toggle Rate = 0.164
```

The following is an example of a state dependent leakage power calculation report.

```
pt_shell> report_power_calculation [get_cell U5] -state "A B"

****************************************
Report : power_calculation
         U5
         -state_condition A B
Design : rpc
Version: W-2004.12
Date   : Thu Sep  2 12:00:48 2004
****************************************

Library(s) Used:

    example (File: /root/libraries/example.db)

Operating Conditions: typical   Library: example
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
rpc                    a                 example


Global Operating Voltage = 0.9
Library Power-specific unit information :
    Voltage Unit : 1 V
    Capacitance Unit : 1 pF
    Time Unit : 1 ns
    Dynamic Power Unit : 0.001 W    (derived from V,C,T units)
    Leakage Power Unit : 1e-09 W

Note: Unless specified, the numbers reported are in library units.

============================================================================
=======
Cell Leakage Power Calculation
    cell: U5
    state condition: A B

  State Dependent Leakage Power = 3.184e-12 W
    cell leakage power = leakage power value * state probability

    Leakage Power Value = 1.021e-07
      library model: scalar
      value = 1.021e-07
      (no scaling since the library has no leakage power k-factors)

    State Probability = 0.03120   (estimated)
```

The following shows how switching power calculation is performed.

```
pt_shell> report_power_calculation [get_net q]
```

Feedback

```
****************************************
Report : power_calculation
         q
Design : rpc
Version: W-2004.12
Date   : Thu Sep  2 12:12:32 2004
****************************************


Library(s) Used:

    example (File: /root/libraries/example.db)

Operating Conditions: typical   Library: example
Wire Load Model Mode: segmented

Design        Wire Load Model            Library
------------------------------------------------
rpc                    a                 example

Global Operating Voltage = 0.9
Library Power-specific unit information :
    Voltage Unit : 1 V
    Capacitance Unit : 1 pF
    Time Unit : 1 ns
    Dynamic Power Unit : 0.001 W    (derived from V,C,T units)
    Leakage Power Unit : 1e-09 W

Note: Unless specified, the numbers reported are in library units.

================================================================================
=======

Net Switching Power Calculation
    net: q
    driver: U4/Y

  Switching power = 1.296e-04 W
    net switching power = switching energy * net toggle rate

  Switching Energy Per Transition = 0.162
    switching energy = 0.5 * capacitance * voltage ^ 2
    total net capacitance = 0.4
    voltage = 0.9

  Net Toggle Rate = 0.8 (user annotated)
```

See Also

- read_saif
- report_lib

- report_power
- set_current_power_net
- set_switching_activity
- update_power
- power_analysis_mode

---

report_power_capacitance

Displays the total power capacitance including wire cap, sum of all sink pin cap and subtract port cap for the specified nets in design, by default reports all nets in design.

Syntax

integer *report_power_capacitance*

```
[-nets object_list]
[-detailed]
[-verbose]
```

Data Types

object_list	list

Arguments

`-nets object_list`

reports total power capacitance of specified nets in design along with wire cap and sum of all sink pin cap. Also reports cap type for each net.

`-detailed`

Generates detailed report for each net separate wire cap and each of the sink pin cap.

`-verbose`

Generates detailed report for each net separate wire cap and each of the sink pin cap. Also provides calculation details for each pin individual looked up cap pieces c1cn aggregation of cap pieces to calculate the lumped capacitance.

object_list

Specifies the objects for which the power capacitance is reported. Objects includes all nets in design, by default reports all nets in design.

Feedback

Description

The *report_power_capacitance* command to query native power capacitance information for the specified net. You can use this information for debugging or verifying power capacitance.

Examples

This example shows net based power capacitance reporting.

```
pt_shell> report_power_capacitance -nets [get_nets *]
****************************************
Report : report_power_capacitance
         -nets
Design : testcase
Version: S-2021.06-DEV-20201111
Date   : Thu Nov 12 12:38:47 2020
****************************************

--------------------------------------------------------------------------------
-------
Net                           WireCap  PinCap   PortCap      TotalPowerCap
 PinCapType
--------------------------------------------------------------------------------
-----------------
IN                             0.0000   0.0131   0.0000       0.0131
 native_c1cn
w                              0.0159   0.0080   0.0000       0.0239
 native_c1cn
OUT                            0.0000   0.0000   0.0000       0.0000
 default
--------------------------------------------------------------------------------
-----------------
```

The following example shows detailed report for each net separate wire cap and each of the sink pin cap.

```
pt_shell> report_power_capacitance -detailed [get_nets IN]
****************************************
Report : report_power_capacitance
         -detailed
         IN
Design : testcase
Version: S-2021.06-DEV-20201111
Date   : Thu Nov 12 12:45:10 2020
****************************************

Library(s) Used:

    tcbn05_bwph210l6p51cnod_base_lvt_ffgnp_0p750v_85c_typical
 (File: /remote/dept5116c/testdata/libraries/power/testc1cn.db)

    Capacitance Unit : 1 pF
```

```
================================================================================
=======

Net capacitance calculation
   net: IN

  Net cap type = native_c1cn
  Net capacitance =  0.0114
    Net capacitance = Wire capacitance + Total pin capacitance

    Wire capacitance =    0

    Port capacitance =       0

    Total pin capacitance =  0.0114

      Total pin capacitance = Sum of individual pin capacitance

      Pin : IN(out)
      Individual pin capacitance   =       0

      Pin : i1/I
      Individual pin capacitance   =  0.0114

================================================================================
=======
```

The following example provides calculation details for each pin of individual looked up cap to calculate the lumped capacitance.

```
pt_shell> report_power_capacitance -verbose IN
****************************************
Report : report_power_capacitance
        -verbose
        IN
Design : testcase
Version: S-2021.06-DEV-20201111
Date   : Thu Nov 12 16:39:59 2020
****************************************

Library(s) Used:

    tcbn05_bwph210l6p51cnod_base_lvt_ffgnp_0p750v_85c_typical
 (File: /remote/dept5116c/testdata/libraries/power/testc1cn.db)

    Capacitance Unit : 1 pF

================================================================================
=======

Net capacitance calculation
   net: IN
```

```
 Net cap type = native_c1cn
 Net capacitance = 0.004724
   Net capacitance = Wire capacitance + Total pin capacitance

   Wire capacitance =    0

   Port capacitance =       0

   Total pin capacitance = 0.004724

     Total pin capacitance = Sum of individual pin capacitance

     Pin : IN(out)
     Individual pin capacitance   =       0
     Individual pin capacitance   = Average of rise and fall
      Average of rise :   0 and fall :   0  =    0

     Pin : i1/I
     Individual pin capacitance   = 0.004724
     rise_slew = 0.005100, fall_slew = 0.005100
     voltage = 0.750000, num_risecap_segments = 9, num_fallcap_segments
= 9
     receiver cap rise threshold pct = [ 0.000000 0.100000 0.300000
0.500000 0.600000 0.700000 0.800000 0.900000 0.999900 1.000000 ]
     receiver cap fall threshold pct = [ 1.000000 0.900000 0.700000
0.500000 0.400000 0.300000 0.200000 0.100000 0.000100 0.000000 ]

     rise capacitance voltage = [0.000000 0.075000 0.225000 0.375000
0.450000 0.525000 0.600000 0.675000 0.749925 0.750000]
     rise capacitance = [0.001482 0.002898 0.004536 0.005126 0.005210
0.004819 0.004429 0.011290 0.002500]
     Weighted C1CN rise cap = 0.004721
     Weighted C1CN rise cap = Sum of weighted C1CN rise cap/Sum of all
weights of rise cap

      Sum of weighted C1CN rise cap = [(Weight * C1CN rise cap)+...]
      Weighted C1CN rise cap =
[(0.075*0.00148)+(0.15*0.0029)+(0.15*0.00454)+(0.075*0.00513)+(0.075*0.0
0521)+(0.075*0.00482)+(0.075*0.00443)+(0.075*0.0113)+(7.5e-05*0.0025)]
      Sum of weighted C1CN rise cap = 0.003541
      Sum of all weights of rise cap = 0.750000

     fall capacitance voltage = [0.750000 0.675000 0.525000 0.375000
0.300000 0.225000 0.150000 0.075000 0.000075 0.000000]
     fall capacitance = [0.001441 0.002897 0.004525 0.005107 0.005178
0.004754 0.004446 0.011509 0.007494]
     Weighted C1CN fall cap = 0.004727
     Weighted C1CN fall cap = Sum of weighted C1CN fall cap/Sum of all
weights of fall cap

      Sum of weighted C1CN fall cap = [(Weight * C1CN fall cap)+...]
```

```
       Sum of weighted C1CN fall cap =
 [(0.075*0.00144)+(0.15*0.0029)+(0.15*0.00452)+(0.075*0.00511)+(0.075*0.0
0518)+(0.075*0.00475)+(0.075*0.00445)+(0.075*0.0115)+(7.5e-05*0.00749)]
       Sum of weighted C1CN fall cap = 0.003546
       Sum of all weights of fall cap  = 0.750000

      Individual pin capacitance   = Average of rise and fall
       Average of rise : 0.004721 and fall : 0.004727  = 0.004724

================================================================================
=======
```

See Also

- report_power_calculation

---

report_power_check_attributes

Allows user to see what bit value represented by option pin_propagation_check, pin_qm_propagation_check, internal_power_check, leakage_power_check.

Syntax

```
string report_power_check_attributes
[-pin_propagation_check]
[-pin_qm_propagation_check]
[-internal_power_check]
[-leakage_power_check]
[-all]
```

Arguments

`-pin_propagation_check`

This option must be provided for attribute pin_propagation_check to know which bit represent what value.

`-pin_qm_propagation_check`

This option must be provided for attribute pin_qm_propagation_check to know which bit represent what value.

`-internal_power_check`

This option must be provided for attribute internal_power_check to know which bit represent what value.

`-leakage_power_check`

This option must be provided for attribute leakage_power_check to know which bit represent what value.

`-all`

This option must be provided for attribute pin_propagation_check , pin_qm_propagation_check , internal_power_check and leakage_power_check to know which bit represent what value.

Description

This command enables user to know bit position what they signify from pin_propagation_check, pin_qm_propagation_check , internal_power_check , leakage_power_check or all attributes.

Examples

The following *report_power_check_attributes command dsipaly option pin_propagation_check bit position what it signify.*

```
pt_shell> report_power_check_attributes -pin_propagation_check

bit sequence            type
left to right
-----------------------------------------
0                       unattempted
1                       missing_lib
2                       missing_function
3                       broken_function
4                       missing_statetable
5                       broken_statetable
6                       clean
```

The following *report_power_check_attributes command display option internal_power_check bit position what it signify.*

```
pt_shell> report_power_check_attributes -internal_power_check

bit sequence            type
left to right
-------------------------------------
0                       unattempted
1                       missing_lib
2                       missing_table
3                       broken_table
4                       out_of_range
5                       clean
```

See Also

- set_power_check_attributes

Feedback

report_power_delay_shifted_event_analysis_options

Reports the options for delay shifted event analysis.

Syntax

integer *report_power_delay_shifted_event_analysis_options*

Description

This command is used when *power_enable_delay_shifted_event_analysis* is set to true. The command is used to report the delay shifted event analysis options set by '*set_power_delay_shifted_event_analysis_options*.

Examples

```
pwr_shell> report_power_delay_shifted_event_analysis_options
effort_level              : low
write_activity_file       : my_design_shifted.fsdb
include_external_delay    : false
```

See Also

- power_enable_delay_shifted_event_analysis
- set_power_delay_shifted_event_analysis_options
- reset_power_delay_shifted_event_analysis_options

report_power_derate

Reports power derate factors annotated on the current design.

Syntax

```
int report_power_derate
[-include_inherited]
[-significant_digits digits]
[-nosplit]
[-x_transition]
[-pin_switching]
[-cnod]
object_list
```

DATA TYPES

```
object_list          list
digits               int
```

Feedback

Arguments

`-include_inherited`

Indicates that the power derating factors reported on each object should also include those set by other objects in the design.

`-significant_digits` *`digits`*

Specifies the number of digits after the decimal point to be displayed for values in the report. Allowed values are 0-12; the default is determined by the report_default_significant_digits variable. Use this option if you want to override the default.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The -nosplit option prevents line-splitting and facilitates writing software to extract information from the report output.

`-x_transition`

Reports representative nets and pins having x-transition power derating factors along with their derate values.

`-cnod`

Reports derate factors applied in context leakage power calculations.

`-pin_switching`

Reports representative pins having pin switching power derating factors along with their derate values.

`object_list`

Specifies current design or a list of cells or library cells or nets or pins from which the power derating factors are reported.

Description

This command is used to report the derate factors either on the design, specific cells, library cells, nets or pins contained in the design. If this command is called without any option, a separate report will be invoked for the power derating factors stored on the current design (referred to as global power derating factors) and also for each cell, library cell, pin or net that has power derating factors set on it.

If this command is called with the boolean option *-include_inherited*, each row containing an inherited derating factor will contain a column entry indicating the object from which it was inherited. If the command is called with the *object_list* specified then derate reports will only be issued for the instances specified in the object list.

For significant digits used in power derate report, the *-significant_digits* option can be used. If the option is not specified, the value of the *report_default_significant_digits* variable is applied. By default, the number of significant digits used to display values is 2.

Examples

In the following example, power derating factors have been set globally on the design; therefore, only global derate factors are reported.

```
pt_shell> set_power_derate -switching 0.92
pt_shell> set_power_derate -internal  1.15
pt_shell> set_power_derate -gate_leakage 0.75
pt_shell> set_power_derate -subthreshold 1.26
pt_shell> report_power_derate

pt_shell> report_power_derate
****************************************
Report : power derate
Design : top
Version: D-2009.12
Date   : Mon Sep 25 11:08:59 2009
****************************************


 Subthreshold    Gate
    Object Name    Object Type     Switching   Internal    Leakage
 Leakage         Leakage
 ------------------------------------------------------------------------
----------------------
    top            design          0.92        1.15        --
 1.26            0.75
1
```

In the following example power derating factors have been set on the design named 'top'. More restrictive derates have been set on the hierarchical block name 'H1'. The derates that apply to the hierarchical cell named, H1/U1, are reported.

```
pt_shell> set_power_derate 0.95
pt_shell> set_power_derate -switching  1.06
pt_shell> set_power_derate -internal -gate_leakage 0.82 [get_cell H1]
pt_shell> set_power_derate -subthreshold_leakage 1.12 [get_cell H1/U1]
pt_shell> report_power_derate -include_inherited [get_cells H1/U1]

****************************************
Report : power derate
        -include_inherited
Design : top
Version: D-2009.12
Date   : Mon Sep 25 11:13:40 2009
****************************************
```

```
 Subthreshold    Gate
  Object Name    Object Type      Switching   Internal    Leakage
 Leakage         Leakage
--------------------------------------------------------------------------
--------------------
  H1/U1          cell (leaf)      1.06        0.82        --
 1.12            0.82
  Inherited                       top         H1          H1           --
             H1
1
```

In the following example x-transition power derating factors have been set on a list of nets, so here representative nets along with their derate values are reported.

```
pt_shell> set_power_derate 0.75 -x_transition {U0/n5 U0/CP U0/Q U0/n6}
pt_shell> report_power_derate -x_transition

****************************************
Report : power derate
Design : top
Version: P-2019.03-DEV-20190129
Date   : Wed Jan 30 04:50:28 2019
****************************************

    Object Name    Object Type           X-transition
 ------------------------------------------------
    U0/n5          net                   0.75
    U0/n6          net                   0.75
    clk            net                   0.75
    out            net                   0.75
1

pt_shell> set_power_derate -pin_switching 1.5 inv1/A
pt_shell> set_power_derate -pin_switching 0.5 {inv2/A reg/D}
pt_shell> report_power_derate -pin_switching

****************************************
Report : power derate
Design : my_design
Version: R-2020.09-SP5-2-DEV-20211208
Date   : Wed Dec  8 17:18:27 2021
****************************************

    Object Name    Object Type           Pin Switching
 ------------------------------------------------
    inv1/A         pin                   1.50
    inv2/A         pin                   0.50
    reg/D          pin                   0.50
1
```

Feedback

See Also

- reset_power_derate
- set_power_derate

---

report_power_domain

Reports the specified power domain.

Syntax

integer *report_power_domain*

[*domain_list*]

Data Types

domain_list	list

Arguments

domain_list

Specifies the list of power domains reported. The names in the list should be a simple (nonhierarchical) name.

If there is no such a power domain with the same name in the current scope, the command fails.

If this option is not specified, all power domain in the specified scope is reported.

Description

The *report_power_domain* command enables you to get the detailed information of an existing power domain in the current scope. The information of the power domain includes the full name, scope, element, and primary power and ground net.

The command returns 1 if it succeeds, and returns 0 otherwise.

Examples

Here are examples of the *report_power_domain* output:

```
pt_shell> report_power_domain
****************************************
Report : power domains
Design : top
Version: A-2007.12
Date   : Thu Sep 27 18:59:04 2007
****************************************
```

```
  Total of 2 power domains defined for design 'top'.

    Power Domain : top_domain
    Scope : top level
    Elements : top level

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>

 ------------------------------------------------------------------------
----

    Power Domain : PD0
    Scope : <top level>
    Elements :  PD0_INST

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>

 ------------------------------------------------------------------------
----

1

pt_shell> report_power_domain [get_power_domain top_domain]
****************************************
Report : power domains
Design : top
Version: A-2007.12-Beta2-DEV
Date   : Thu Sep 27 18:59:04 2007
****************************************

    Power Domain : top_domain
    Scope : <top level>
    Elements : <top level>

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>

 ------------------------------------------------------------------------
----

1

pt_shell> report_power_domain [get_power_domain -regexp {t.*}]
****************************************
Report : power domains
Design : top
Version: A-2007.12
Date   : Thu Sep 27 18:59:04 2007
****************************************

    Power Domain : top_domain
```

```
    Scope : <top level>
    Elements : <top level>

    Connections :       -- Power --           -- Ground --
    Primary             <none>                <none>

 ------------------------------------------------------------------------
----

1
```

See Also

- create_power_domain
- get_power_domains

---

report_power_groups

Report the existing power groups.

Syntax

string *report_power_groups*

```
[group_names]
[-nosplit]
```

Data Types

```
group_names        list
```

Arguments

group_names

Specifies the power group names that are to be reported.

-nosplit

Indicates to prevent line splitting if column overflows.

Description

The *report_power_groups* command reports the power groups that has been created in the earlier stage, including the predefined power groups.

Examples

In the following example, all the existing power groups are reported.

```
pt_shell> create_power_group -name clock_tree [get_clock_network_objects
 -type cell]
```

Feedback

```
1
pt_shell> report_power_groups

i*************************************
Report : report_power_groups
         -nosplit
Design : testcase
Version: Y-2006.06-Beta1-DEV
Date   : Mon Mar 13 16:28:48 2006
*************************************

  Power Group                              Size           Attribute
-------------------------------------------------------------------
black_box                                     0       Default
clock_network                                 5       Default
combinational                                 0       Default
io_pad                                        0       Default
memory                                        0       Default
register                                     25       Default
sequential                                    0       Default
clock_tree                                    5       User defined

1
```

See Also

- create_power_group
- remove_power_groups
- get_power_group_objects

---

report_power_net_info

Reports the power net info for the current design.

Syntax

int *report_power_net_info*

Arguments

None.

Description

The *report_power_net_info* command reports the power net info for the current design.

Examples

The following example uses the command to report the power net info.

Feedback

```
pt_shell> create_power_domain T
1
pt_shell> create_power_net_info T_VDD -power
1
pt_shell> create_power_net_info T_VSS -gnd
1
pt_shell> create_power_domain T
1
pt_shell> connect_power_domain T \\
          -primary_power_net T_VDD \\
          -primary_ground_net T_VSS
1
prompt> report_power_net_info

****************************************
Report : power_net
Design : top
Version: Z-2007.06-DEV
Date   : Wed May 16 15:18:42 2007
****************************************

  Total of 2 power nets defined.

  Power Net 'T_VDD' (power)
  -------------------------------------------------
  Primary Power Hookups:                          T
  Internal Power Hookups:                         T
  -------------------------------------------------

  Power Net 'T_VSS' (ground)
  -------------------------------------------------
  Primary Ground Hookups:                         T
  Internal Ground Hookups:                        T
  -------------------------------------------------

1
```

See Also

- create_power_net_info
- connect_power_domain

---

report_power_network

Reports connectivity in the virtual power network formed by UPF supply nets, ports, and PG pins.

Syntax

string *report_power_network*

```
[-nets nets]
```

Data Types

```
nets      string
```

Arguments

```
-nets nets
```

Reports explicit connectivity of a subset of nets.

Description

This command reports the explicit connectivity of supply nets, ports, and connected power and ground pins.

Examples

The following example shows the explicit connections of the given supply net, VDDI when you specify the *connect_supply_net* command.

```
prompt> report_power_network -nets VDDI

 Supply Net: VDDI
  Explicit Connections:
    Name :              Object Type           Domain
------------------------------------------------------------------------------
------
    VDDI                Supply Port           TOP
    InstDecode/LS_u1/VDDL PG_power_pin        INST
in -0.25in
```

See Also

- create_supply_net
- report_supply_net

---

report_power_pin_info

Reports the power pin information for logic library cells or leaf cells.

Syntax

status *report_power_pin_info*

```
        object_list
```

Feedback

Data Types

```
object_list      list
-show_rail_name
```

Arguments

object_list

Specifies a list of logic library cells or a list of leaf cells or ports.

-show_rail_name

this option gives the rail name.

Description

The *report_power_pin_info* command reports the power pin information for logic library cells or leaf level cells in the current design.

If you specify a list of library cells, the name, the types, and the voltage specification of the power pins are reported. All types of PG pins including power, ground, and bias pins are shown.

If you specify a list of leaf cells as the *object_list*, the power net information that are connected to the power pins are also reported. Hierarchical cells are ignored by this command.

Examples

The following example uses the command to report the power pin information.

```
pt_shell> report_power_pin_info [get_cells -hierarchical]

****************************************
Report : power pin info
Design : top
 ...
****************************************

Note: Power connections marked by (*) are exceptional

Cell            Power Pin Name      Type                Voltage   Power
 Net Connected
-------------------------------------------------------------------------
---------------
PD0_INST/I0     PWR                 primary_power        1.0000
PD0_INST/I0     GND                 primary_ground       0.0000
PD0_INST/I1     PWR                 primary_power        1.0000
PD0_INST/I1     GND                 primary_ground       0.0000
I0              PWR                 primary_power        1.0000
I0              GND                 primary_ground       0.0000
I1              PWR                 primary_power        1.0000
```

```
I1              GND                  primary_ground       0.0000
1

pt_shell> report_power_pin_info [get_lib_cells -of_objects [get_cells
 -hierarchical]]

****************************************
Report : power pin info
Design : top
 ...
****************************************

Library Cell    Power Pin Name       Type                 Voltage
-----------------------------------------------------------------
INVX2           PWR                  primary_power         1.0000
INVX2           GND                  primary_ground        0.0000
BUFX2           PWR                  primary_power         1.0000
BUFX2           GND                  primary_ground        0.0000
AND2X1          PWR                  primary_power         1.0000
AND2X1          GND                  primary_ground        0.0000
1
pt_shell> connect_power_net_info I0 -power_pin_name PWR -power_net_name
 exp_VDD
1
pt_shell> connect_power_net_info PD0_INST/I0 -power_pin_name PWR
 -power_net_name exp_VDD
1
pt_shell> report_power_pin_info [get_cells -hierarchical]

****************************************
Report : power pin info
Design : top
 ...
****************************************

Note: Power connections marked by (*) are exceptional

Cell            Power Pin Name       Type                 Voltage   Power
 Net Connected
-----------------------------------------------------------------------------
-----------
PD0_INST/I0     PWR                  primary_power         1.0000   exp_VDD
 (*)
PD0_INST/I0     GND                  primary_ground        0.0000   A_VSS
PD0_INST/I1     PWR                  primary_power         1.0000   A_VDD
PD0_INST/I1     GND                  primary_ground        0.0000   A_VSS
I0              PWR                  primary_power         1.0000   exp_VDD
 (*)
I0              GND                  primary_ground        0.0000   T_VSS
I1              PWR                  primary_power         1.0000   T_VDD
I1              GND                  primary_ground        0.0000   T_VSS
1
```

Feedback

```
pt_shell> report_power_pin_info [get_cells -hierarchical] -show_rail_name

****************************************
Report : power pin info
Design : top
 ...
****************************************

Note: Power connections marked by (*) are exceptional

Cell            Power Pin Name      Type                Voltage   Power
 Net Connected     Rail Name
-------------------------------------------------------------------------
----------------------------
PD0_INST/I0     PWR                 primary_power        1.0000   exp_VDD
 (*)                PWR
PD0_INST/I0     GND                 primary_ground       0.0000   A_VSS
                   GND
PD0_INST/I1     PWR                 primary_power        1.0000   A_VDD
                   PWR
PD0_INST/I1     GND                 primary_ground       0.0000   A_VSS
                   GND
I0              PWR                 primary_power        1.0000   exp_VDD
 (*)                PWR
I0              GND                 primary_ground       0.0000   T_VSS
                   GND
I1              PWR                 primary_power        1.0000   T_VDD
                   PWR
I1              GND                 primary_ground       0.0000   T_VSS
                   GND
1
```

See Also

- connect_power_domain
- connect_power_net_info

---

report_power_rail_mapping

Reports the current power rail mapping.

Syntax

status *report_power_rail_mapping*

```
[-cells cell_list]
[-verbose]
[-nosplit]
```

Feedback

Data Types

cell_list	list

Arguments

`-cells` *cell_list*

Specifies the instance names or collections of instances. This only takes effect when the *-verbose* option is used. The instance can be hierarchical or leaf instances. For a instance block, only the top instance needs to be specified. All the leaf instances inside such blocks are reported. Without this option, the mapping is reported for the whole design.

`-verbose`

Reports detailed rail mapping for each individual leaf instance. Choose the leaf instances to be reported by using the *-instance* option. When this option is used, PrimePower also checks potential problems in rail mapping done for the instances. The list of checks are:

- Different library rails are mapped to the same design rail in a multirail leaf cell
- No rail specific power table is defined for a library cell with multiple power rails

`-nosplit`

Most of the information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column. This option prevents line-splitting and facilitates writing software to extract information from the report output.

Description

The *report_power_rail_mapping* command displays all the power rails defined in the library and those existing in the design. The library power rails are the *voltage_map* attributes defined in the library (or the *power_rail* attributes defined in the *power_supply* group). The design power rails are the rails created by the *create_power_rail_mapping* command. If a library does not have rail information, default_rail is shown for its library power rail. If the mapping does not exist, the default design power rail is shown and labeled with *(default)*. During power simulation, those missing cases are actually mapped to the default design power rail. Internally, there is a default rail associated with each library and with the whole design. If the *-verbose* option is used, PrimePower reports the detailed rail mapping for each individual leaf instances and also shows possible rail mapping problems for design.

Examples

Here is an example of the *report_power_rail_mapping* command:

```
pt_shell> report_power_rail_mapping -verbose -cells block1
```

Feedback

See Also

- create_power_rail_mapping

---

report_power_switch

Report all power switches defined in the design.

Syntax

int *report_power_switch*

Arguments

None.

Description

The *report_power_switch* command enables you to get the detail information of all power_switches defined in the design. The power switch information includes:

- full name
- power domain where it is created in
- output supply port
- input supply port
- control port
- on_state statement

The command returns 1 if it succeed; otherwise, it returns 0.

Examples

The following example reports the information of all power_switches defined:

```
pt_shell> report_power_switch

  Power Switch : SW1
----------------------------------------
    Power Domain : PD1
    Output Supply Port : vout VN2
    Input Supply Port : vin VN1
    Control Port : ctrl_small ON1
    Control Port : ctrl_large ON2
    On State : state1 vin { ON1 & ON2 }
----------------------------------------
1
```

See Also

- create_power_switch

Feedback

report_pst

Reports the power state table defined in the current design previously using *add_power_state* command.

Syntax

status *report_pst*

```
[-verbose]
```

Arguments

```
-verbose
```

Also reports the power states defined on supply sets.

Description

This command reports the power state table of the current design. This reflects the power state table defined by user using *add_power_state* and *create_power_state_group* commands.

Examples

The following UPF script defines the power states on the supplies and then the legal power state combinations for the design, and the *report_pst* command reports the resulting power state table.

```
prompt> add_power_state -supply SS1 \\
        -state ON  {-supply_expr {power == {FULL_ON 0.8} && ground ==
 {FULL_ON 0}}}
prompt> add_power_state -supply SS2 \\
        -state ON  {-supply_expr {power == {FULL_ON 0.8} && ground ==
 {FULL_ON 0}}} \\
        -state OFF {-supply_expr {power == {OFF} && ground == {FULL_ON
 0}}}
prompt> create_power_state_group MY_PST_GROUP
prompt> add_power_state -group MY_PST_GROUP \\
        -state RUN   {-logic_expr {SS1 == ON && SS2 == ON}} \\
        -state SLEEP {-logic_expr {SS1 == ON && SS2 == OFF}}

prompt> report_pst
 ...

  Total of 1 power state group defined for design 'mydesign'.

------------------------------------------------------------------------
------
    Power State Group : MY_PST_GROUP
      Scope : <top level>
      Group Members : SS1          SS2
```

```
      Total of 2 power state combinations on this group :
      RUN           : ON           ON
      SLEEP         : ON           OFF

prompt>report_pst -verbose
 ...

  Total of 1 power state group defined for design 'mydesign'.

--------------------------------------------------------------------------
------
    Power State Group : MY_PST_GROUP
      Scope : <top level>
      Group Members : SS1          SS2
      Total of 2 power state combinations on this group :
      RUN           : ON           ON
      SLEEP         : ON           OFF


  Power states defined on supply sets for design 'top'.

--------------------------------------------------------------------------
------
    Supply Set : SS1 has 1 state defined:
      State : ON, power : FULL_ON 0.90, ground : FULL_ON 0.00
--------------------------------------------------------------------------
------
    Supply Set : SS2 has 2 states defined:
      State : ON, power : FULL_ON 0.90, ground : FULL_ON 0.00
      State : OFF, power : OFF, ground : FULL_ON 0.00
```

See Also

- add_power_state
- create_power_state_group

---

report_pulse_clock_max_transition

Displays maximum transition computation at the input of pulse generator and pulse clock network.

Syntax

int *report_pulse_clock_max_transition*

```
[-all_violators]
[-significant_digits digits]
[-nosplit]
[-rise]
[-fall]
```

Feedback

```
[-transitive_fanout]
[-sms_scenarios sms_scenarios_list]
[port_pin_list]
```

Data Types

digits	int
port_pin_list	list
sms_scenarios_list	collection

Arguments

`-all_violators`

Specifies that only the pins violating maximum transition constraint are reported.

`-significant_digits` *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which defaults to 2. Use this option if you want to override the default.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

`-rise`

Report slack for rise transition.

`-fall`

Report slack for fall transition.

`-transitive_fanout`

Report transition slack at the transitive fanout of pulse generator cells. If this option is not set, the transition slack is reported at the input of pulse generator cells.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock maximum transition constraint.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only pulse clock information compatible to the specified SMS scenarios collection is reported. This option

is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *report_pulse_clock_max_transition* command displays the maximum transition computation of pins and ports. By default, only the input pins of the pulse generators are reported. If rise or fall is not specified, both rise and fall transition slack are reported. To enable pulse clock constraint checking, ensure the *timing_enable_pulse_clock_constraints* variable is set to *true*, which is its default.

Examples

The following example generates a report of all transition slack computation at the input of pulse generators in the current design.

```
pt_shell> report_pulse_clock_max_transition
****************************************
Report : pulse clock max transition
        -rise
        -fall
Design : test
****************************************

   pulse_clock_max_transition_rise

                            Required        Actual
   Pin                     Transition     Transition        Slack
   ------------------------------------------------------------------
   p1/i                        0.40           0.12          0.28   (MET)

   pulse_clock_max_transition_fall

                            Required        Actual
   Pin                     Transition     Transition        Slack
   ------------------------------------------------------------------
   p1/i                        0.40           0.12          0.28   (MET)
```

The following example generates a report of all transition slack computation in the pulse clock network in the current design.

```
pt_shell> report_pulse_clock_max_transition -transitive_fanout
****************************************
Report : pulse clock max transition
        -rise
        -fall
Design : test
****************************************

   pulse_clock_max_transition_rise_transitive_fanout

                            Required        Actual
```

```
Pin                          Transition     Transition        Slack
-------------------------------------------------------------------
ff2/cp                          0.20           0.08           0.12   (MET)
ff1/cp                          0.20           0.02           0.18   (MET)
u3/zn                           0.30           0.08           0.22   (MET)
p1/z                            0.30           0.02           0.28   (MET)
u3/i                            0.30           0.02           0.28   (MET)

pulse_clock_max_transition_fall_transitive_fanout

                             Required       Actual
Pin                          Transition     Transition        Slack
-------------------------------------------------------------------
ff2/cp                          0.20           0.08           0.12   (MET)
ff1/cp                          0.20           0.02           0.18   (MET)
u3/zn                           0.30           0.08           0.22   (MET)
p1/z                            0.30           0.02           0.28   (MET)
u3/i                            0.30           0.02           0.28   (MET)
```

See Also

- set_pulse_clock_max_transition
- remove_pulse_clock_max_transition
- report_constraint

---

report_pulse_clock_max_width

Displays maximum pulse width computation for the pulse clock network.

Syntax

int *report_pulse_clock_max_width*

```
[-all_violators]
[-significant_digits digits]
[-nosplit]
[-path_type format]
[-transitive_fanout]
[-sms_scenarios sms_scenarios_list]
[port_pin_list]
```

Data Types

```
digits                int
format                string
port_pin_list         list
sms_scenarios_list    collection
```

Feedback

Arguments

`-all_violators`

Specifies that only the pins in the pulse clock network violating the maximum pulse width constraint are reported.

`-significant_digits` *`digits`*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which defaults to 2. Use this option if you want to override the default.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

`-path_type` *`format`*

Specifies the format of the path report and how the clock path is displayed. The allowed values are its default of *summary* and also *short*. The *summary* value generates a report with a column format that shows one line for each path and shows only the required pulse width, actual pulse width and slack; The *short* value displays only startpoints and endpoints in the clock path.

`-transitive_fanout`

Report constraints at the transitive fanout of pulse generators. Currently, specifying or not specifying this option has the same behavior.

`port_pin_list`

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock maximum width constraint.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only pulse clock information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *report_pulse_clock_max_width* command displays the maximum pulse clock width computation of the specified pins and ports. By default all pins with pulse clock width constraint are reported. To enable pulse clock constraint checking, ensure that the *timing_enable_pulse_clock_constraints* variable is set to *true*, which is the default.

Feedback

Examples

The following example generates a report of all maximum pulse width violations in the current design.

```
pt_shell> report_pulse_clock_max_width
****************************************
Report : pulse clock max width
         -path_type summary
Design : test
****************************************

   sequential_pulse_clock_max_width

                       Required       Actual
   Pin                 pulse width    pulse width       Slack
   ------------------------------------------------------------------
   ff1/cp (high)            1.00           0.54          0.46   (MET)
   ff2/cp (low)             1.00           0.56          0.44   (MET)


   clock_tree_pulse_clock_max_width

                       Required       Actual
   Pin                 pulse width    pulse width       Slack
   ------------------------------------------------------------------
   u3/zn (low)              1.00           0.56          0.44   (MET)
   p1/z (high)              1.00           0.54          0.46   (MET)
   u3/i (high)              1.00           0.54          0.46   (MET)
```

See Also

- remove_pulse_clock_max_width
- report_constraint
- set_pulse_clock_max_width
- report_default_significant_digits
- timing_enable_pulse_clock_constraints

---

report_pulse_clock_min_transition

Displays minimum transition computation at the input of pulse generator.

Syntax

int *report_pulse_clock_min_transition*

```
[-all_violators]
[-significant_digits digits]
```

Feedback

```
[-nosplit]
[-rise]
[-fall]
[-sms_scenarios sms_scenarios_list]
[port_pin_list]
```

Data Types

```
digits                int
port_pin_list         list
sms_scenarios_list    collection
```

Arguments

`-all_violators`

Specifies that only the input pins of pulse generators violating minimum transition constraint is reported.

`-significant_digits digits`

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which defaults to 2. Use this option if you want to override the default.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

`-rise`

Reports slack for rise transition.

`-fall`

Reports slack for fall transition.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock minimum transition constraint.

`-sms_scenarios sms_scenarios_list`

Specifies a collection of SMS scenarios to analyze. Only pulse clock information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Feedback

Description

The *report_pulse_clock_min_transition* command displays the minimum transition computation of pins and ports. By default, all pins with pulse clock minimum transition constraint are reported. Similarly, if rise or fall is not specified, then both rise and fall transition slack is reported. To enable pulse clock constraint checking, ensure the *timing_enable_pulse_clock_constraints* variable is set to its default of *true*.

Examples

The following example generates a report of all minimum transition violations in the current design.

```
pt_shell> report_pulse_clock_min_transition
****************************************
Report : pulse clock min transition
         -rise
         -fall
Design : test
****************************************

   pulse_clock_min_transition_rise

                             Required        Actual
   Pin                      Transition     Transition        Slack
   ------------------------------------------------------------------
   p1/i                          0.30           0.12          -0.18
 (VIOLATED)

   pulse_clock_min_transition_fall

                             Required        Actual
   Pin                      Transition     Transition        Slack
   ------------------------------------------------------------------
   p1/i                          0.30           0.12          -0.18
 (VIOLATED)
```

See Also

- set_pulse_clock_min_transition
- remove_pulse_clock_min_transition
- report_constraint

---

report_pulse_clock_min_width

Displays minimum pulse width computation for the pulse clock network.

Feedback

Syntax

int *report_pulse_clock_min_width*

```
[-all_violators]
[-significant_digits digits]
[-nosplit]
[-path_type format]
[-transitive_fanout]
[-sms_scenarios sms_scenarios_list]
[port_pin_list]
```

Data Types

digits	int
format	string
port_pin_list	list
sms_scenarios_list	collection

Arguments

`-all_violators`

Specifies that only the pins in the pulse clock network violating the minimum pulse width constraint is reported.

`-significant_digits digits`

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, whose default value is 2. Use this option if you want to override the default.

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the width of the column, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line splitting and facilitates writing software to extract information from the report output.

`-path_type format`

Specifies the format of the path report and how the clock path is displayed. Allowed values are *summary* (the default) and *short*. The *summary* value generates a report with a column format that shows one line for each path and shows only the required pulse width, actual pulse width, and slack. The *short* value displays only start and end points in the clock path.

`-transitive_fanout`

Reports constraints at the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

port_pin_list

Specifies a list of pins or ports to report. By default, the report contains all pins with pulse clock minimum width constraint.

-sms_scenarios *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only pulse clock information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *report_pulse_clock_min_width* command displays the minimum pulse clock width computation of the specified pins and ports. By default, all pins with pulse clock width constraint are reported. To enable pulse clock constraint checking, ensure the *timing_enable_pulse_clock_constraints* variable is set to its default of *true*.

Examples

The following example generates a report of all minimum pulse width violations in the current design.

```
pt_shell> report_pulse_clock_min_width
****************************************
Report : pulse clock min width
         -path_type summary
Design : test
****************************************

   sequential_pulse_clock_min_width

                            Required        Actual
   Pin                     pulse width    pulse width       Slack
   ------------------------------------------------------------------
   ff2/cp (low)                2.00           0.56          -1.44
 (VIOLATED)
   ff1/cp (high)               0.80           0.54          -0.26
 (VIOLATED)


   clock_tree_pulse_clock_min_width

                            Required        Actual
   Pin                     pulse width    pulse width       Slack
   ------------------------------------------------------------------
   u3/i (high)                 0.80           0.54          -0.26
 (VIOLATED)
   u3/zn (low)                 0.80           0.56          -0.24
 (VIOLATED)
```

```
   p1/z (high)                    0.80           0.54          -0.26
(VIOLATED)
```

See Also

- set_pulse_clock_min_width
- remove_pulse_clock_min_width
- report_constraint

---

report_pvt_explorer_condition

Reports PVT explorer condition set by *set_pvt_explorer_condition*.

Syntax

string *report_pvt_explorer_condition*

Description

The *report_pvt_explorer_condition* command reports PVT explorer condition set by *set_pvt_explorer_condition*, which is used in PrimeShield PVT Explorer what-if analysis. *define_sensitivity_lib_mapping* is required to do the analysis. And *set ps_enable_pvt_explorer_analysis true* is required to use *set_pvt_explorer_condition* or *report_pvt_explorer_condition* command.

Examples

The following script enables PVT Explorer analysis, and report the PVT explorer condition, where "one_cond.txt" file content as follows:
#index,pmos_lvt_vt,nmos_lvt_vt,pmos_lvt_ids0,nmos_lvt_ids0 1,0.020,0.020,0.15,0.15

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_pvt_explorer_analysis true
set ps_pvt_explorer_pba_only_mode false
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]

set_pvt_explorer_condition one_cond.txt
update_timing -full
update_power

pt_shell> report_pvt_explorer_condition
Information: PVT Explorer condition =
  param_name  perturbation
================================================
  pmos_lvt_vt    0.02
  nmos_lvt_vt    0.02
```

```
  pmos_lvt_ids0  0.15
  nmos_lvt_ids0  0.15

1
```

The following script enables PVT Explorer what-if analysis for timing path robustness, and report the PVT explroer condition, where "multiple_DoEs.txt" file content as follows:
#index,pmos_lvt_vt,nmos_lvt_vt,pmos_lvt_ids0,nmos_lvt_ids0 1,0.020,0.020,0.15,0.15 2,0.011,0.011,0.10,0.10

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_pvt_explorer_analysis true
set ps_pvt_explorer_pba_only_mode true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]
update_timing -full  # baseline QoR
set paths [get_timing_paths ...]

set_pvt_explorer_condition multiple_DoEs.txt
#update original path collection with slack shift per DoE
set paths_new [get_timing_paths -pba_mode path $paths]
#generate slack shift per DoE for all paths
puts [get_attr $paths_mew pvt_explorer_slack_shift]

pt_shell> report_pvt_explorer_condition
Information: PVT Explorer condition loaded as follows
index pmos_lvt_vt nmos_lvt_vt pmos_lvt_ids0 nmos_lvt_ids0
============================================================
1  0.02   0.02   0.15  0.15
2  0.011  0.011  0.10  0.10
============================================================
```

See Also

- define_sensitivity_lib_mapping
- reset_pvt_explorer_condition
- set_pvt_explorer_condition
- ps_enable_spice2design_analysis
- ps_enable_pvt_explorer_analysis
- ps_pvt_explorer_pba_only_mode

---

report_qor

Provides an overview of the quality of a design.

Feedback

Syntax

```
status report_qor
[-max]
[-min]
[-significant_digits digits]
[-only_violated]
[-summary]
[-physical]
[-pba_mode none | path | exhaustive | ml_exhaustive]
[-sms_scenarios sms_scenarios_list]
[-drc_violations]
[-area]
[-object_count]
[-timing]
```

Data Types

```
digits                integer
sms_scenarios_list    collection
```

Arguments

`-max`

Indicates that only maximum delay and setup information is to be displayed. If this option is not specified, the report for both min and max delay types is produced.

`-min`

Indicates that only minimum delay and hold information is to be displayed. If this option is not specified, the report for both min and max delay types is produced.

`-significant_digits` *digits*

Specifies the number of reported digits to the right of the decimal point. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which has a default of 2. Use this option to override the default.

`-only_violated`

Shows timing path group summaries for only violating path groups. The area, object count, and DRC violation sections will be displayed unless the *-timing* is specified.

`-summary`

Summarizes the worst negative slack (WNS) and total negative slack (TNS) for all path groups. See *timing_report_union_tns* for details of TNS calculation. The area, object count, and DRC violation sections will be displayed unless the *-timing* is specified.

`-physical`

This option has no effect in PrimeTime.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Controls path-based analysis with the following modes:

- *none* (the default) - Path-based analysis is not applied.
- *path* - Path-based analysis is applied over the paths that gave the worst graph-based analysis violation. There is no guarantee that the reported path-based analysis slack over every endpoint is the worst one.
- *exhaustive* - An exhaustive path-based analysis path search algorithm is applied to determine the worst path-based analysis slack violations.
- *ml_exhaustive* - Performs Machine Learning based exhaustive path-based analysis. This mechanism will deploy machine learning techniques to trade runtime vs accuracy during the design flow. Accuracy and runtime will match pba_mode exhaustive as the design approaches signoff.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only QoR information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

`-timing`

Print per path group and for setup and hold; the length of the worst path (shown as the "Critical Path Length" in the report, which represents the arrival time of the signal at the path's endpoint) and its slack as well as the the total negative slack.

The area, object count and drc violation sections will not be displayed unless their respective options are also provided.

`-area`

Print the total area summary only

`-object_count`

Print the cell and pin count summary only

`-drc_violations`

Print the summary of design related constraint violations.

Feedback

Description

The *report_qor* command provides an overview of the quality of results (QoR) for a design, including the length of the worst paths (shown as "Critical Path Length" in the report, which represents the arrival time of the signal at the path's endpoint), total negative slack, minimum delay and hold summaries, and a detailed DRC summary. When no options are specified the default output is equivalent to *report_qor -timing -area -object_counts -drc_violations*. Specifying any of *-timing*, *-object_counts*, *-drc_violations*, *-area*, *-summary*, and *-only_violated* options tailors the output to only report those summary sections.

Only *-timing*, *-object_counts*, *-drc_violations*, and *-area* are supported at the manager in distributed analysis mode. All the options are supported with the command, if executed on remote workers as part of the distributed command *remote_execute -partitions*.

If you specify the *-pba_mode exhaustive* option, path-based analysis is used to calculate the slack values. This option requires that a path search be performed to ensure that the returned slack values are truly the worst. The additional runtime required for this option depends on the amount of timing improvement resulting from path-based analysis. As the timing improvement from path-based analysis increases, the runtime for the path search increases.

Examples

The following example shows detailed QoR information for the current design; some timing path group summaries are omitted for clarity:

```
pt_shell> report_qor
****************************************
Report : qor
Design : example
 ...
****************************************

Timing Path Group '**async_default**' (max_delay/setup)
-----------------------------------------
Levels of Logic:                         50
Critical Path Length:                 20.42
Critical Path Slack:                 -23.71
Total Negative Slack:              -1010.10
No. of Violating Paths:                 186
-----------------------------------------

Timing Path Group 'ADC_SCLK_IN' (max_delay/setup)
-----------------------------------------
Levels of Logic:                         20
Critical Path Length:                  6.31
Critical Path Slack:                  -2.10
Total Negative Slack:                 -39.80
No. of Violating Paths:                  32
```

```
--------------------------------------------

Timing Path Group 'CLK1' (max_delay/setup)
--------------------------------------------
Levels of Logic:                         7
Critical Path Length:                 2.62
Critical Path Slack:                  0.26
Total Negative Slack:                 0.00
No. of Violating Paths:                  0
--------------------------------------------

Timing Path Group '**async_default**' (min_delay/hold)
--------------------------------------------
Levels of Logic:                         3
Critical Path Length:                 0.79
Critical Path Slack:                 -3.37
Total Negative Slack:               -13.19
No. of Violating Paths:                  4
--------------------------------------------

Timing Path Group 'ADC_PCMCLK_IN' (min_delay/hold)
--------------------------------------------
Levels of Logic:                         2
Critical Path Length:                 0.84
Critical Path Slack:                 -1.54
Total Negative Slack:               -23.26
No. of Violating Paths:                 33
--------------------------------------------

Timing Path Group 'mem_ctl0/st20_reg_q_regx3x/Q' (min_delay/hold)
--------------------------------------------
Levels of Logic:                         1
Critical Path Length:                 0.42
Critical Path Slack:                  0.38
Total Negative Slack:                 0.00
No. of Violating Paths:                  0
--------------------------------------------

Cell Count
--------------------------------------------
Hierarchical Cell Count:              2310
Hierarchical Port Count:             38125
Leaf Cell Count:                     24230
--------------------------------------------

Area
--------------------------------------------
Design Area:                   5659902.00
--------------------------------------------

Design Rule Violations
--------------------------------------------
Total No. of Pins in Design:         87497
```

```
max_capacitance Count:                    305
max_fanout Count:                         105
clock_gating_setup Count:                  32
clock_gating_hold Count:                   37
max_capacitance Cost:                   -52.87
max_fanout Cost:                      -3202.00
clock_gating_setup Cost:               -409.27
clock_gating_hold Cost:                -138.85
Total DRC Cost:                       -3802.99
-------------------------------------------

pt_shell> report_qor -area
****************************************
Report : qor
Design : example
 ...
****************************************

Area
-------------------------------------------
Design Area:                       5659902.00
-------------------------------------------

pt_shell> report_qor -drc_violations
****************************************
Report : qor
Design : example
 ...
****************************************

Design Rule Violations
-------------------------------------------
Total No. of Pins in Design:            87497
max_capacitance Count:                    305
max_fanout Count:                         105
clock_gating_setup Count:                  32
clock_gating_hold Count:                   37
max_capacitance Cost:                   -52.87
max_fanout Cost:                      -3202.00
clock_gating_setup Cost:               -409.27
clock_gating_hold Cost:                -138.85
Total DRC Cost:                       -3802.99
-------------------------------------------
```

See Also

- report_analysis_coverage
- report_constraint
- report_global_timing

Feedback

- report_default_significant_digits
- timing_report_union_tns

report_qtm_model

Reports quick timing model (QTM) data.

Syntax

string *report_qtm_model*

```
[-global_parameters]
[-ports]
[-arcs]
[-nosplit]
```

Arguments

`-global_parameters`

Reports the global parameters of the current quick timing model.

`-ports`

Reports the ports of the current quick timing model.

`-arcs`

Reports the timing arcs of the current quick timing model.

`-nosplit`

Prevents line splitting and facilitates writing software to extract information from the report output. Most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line starting in the correct column.

Description

The *report_qtm_model* command prints a report of the current quick timing model. By default, the command reports the global parameters, ports, and arcs of the model.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following command reports all the attributes of the currently active quick timing model.

```
pt_shell> report_qtm_model

****************************************
Report : qtm_model
```

```
 ...
****************************************

PATH TYPES
Type     Cell      Input pin   Output pin   Fanout           Delay per
 level
-------------------------------------------------------------------------
-------
path1    AN210     A           Y            2                0.570929

DRIVE TYPES
Type     Cell      Input pin   Output pin
-------------------------------------------------------------------------
-------
drive1   AN210     A           Y

LOAD TYPES
Type     Cell      Input pin      Capacitance
-------------------------------------------------------------------------
-------
load1    AN210     A              1

CLOCK RELATED PARAMETERS
Parameter    Cell      Clock Pin       Related Pin    Delay
-------------------------------------------------------------------------
-------
setup        DTN10     CLK             D              1.33
hold         DTN10     CLK             D              1.33
setup        DTN10     CLK             Q              1.80244

********************************************
Wire load model not defined
********************************************
********************************************
Max Transition not defined
********************************************
PORTS
Port     Direction Width Cap Value  Load Type  Drive Value  Drive Type
-------------------------------------------------------------------------
-------
CLK      CLOCK     1      -           -         -            -
A        INPUT     1      2         load1       -            -
B        INPUT     1      2         load1       -            -
X        OUTPUT    1      -           -         -          drive1
Y        OUTPUT    1      -           -         -          drive1

ARCS
Name     From               To                 Delay  Path Type   Path
 Factor
-------------------------------------------------------------------------
-------
CLK_to_A CLK                A                  1.142  path1        2
CLK_to_B CLK                B                  3.997  path1        7
```

```
CLK_to_X CLK                X                   3.000   -           -
CLK_to_Y CLK                Y                   7.000   -           -
```

The following command reports the global parameters of the currently active quick timing model.

```
pt_shell> report_qtm_model -global_parameters

****************************************
Report : qtm_model
 ...
****************************************

PATH TYPES
Type     Cell      Input pin   Output pin   Fanout           Delay per
 level
-------------------------------------------------------------------------
-------
path1    AN210     A           Y            2                0.570929

DRIVE TYPES
Type     Cell      Input pin   Output pin
-------------------------------------------------------------------------
-------
drive1   AN210     A           Y

LOAD TYPES
Type     Cell      Input pin      Capacitance
-------------------------------------------------------------------------
-------
load1    AN210     A              1

CLOCK RELATED PARAMETERS
Parameter    Cell      Clock Pin       Related Pin    Delay
-------------------------------------------------------------------------
-------
setup        DTN10     CLK             D              1.33
hold         DTN10     CLK             D              1.33
setup        DTN10     CLK             Q              1.80244

*********************************************
Wire load model not defined
*********************************************
*********************************************
Max Transition not defined
*********************************************
```

See Also

- create_qtm_model
- save_qtm_model

Feedback

report_rc_components

Reports the RC contributions of the specified nets.

Syntax

status *report_rc_components*

`-of_objects` *nets*

Data Types

nets	list

Arguments

`-of_objects` *nets*

A list of one or more nets for which to report the RC components. At least one net is required. Glob-style wildcards (*) are supported.

Description

This command reports the RC contributions of the specified nets.

This command is supported only for transistor-level GPDs.

Examples

The following example shows an RC component report. The report contains a section for each specified net.

```
pt_shell> report_rc_components -of_objects min_lsb_led[5]
*********************************
Report : RC Components
Design : toprt
Version: Q-2019.12
Date   : Tue Nov 12 13:23:45 2019
*********************************

Net: min_lsb_led[5]
Total Resistance: 0.152999 kOhms
Total Capacitance: 0.057662 pF

Layer  ResValue(KOhms)  %ResContribution  CapValue(pF)  %CapContribution
=====  ===============  ================  ============  ================
M1     0.003774         2.466683          0.000000      0.000000
M2     0.025163         16.446513         0.010522      18.247719
M3     0.118932         77.733841         0.047140      81.752281
VIA1   0.002250         1.470598          0.000000      0.000000
VIA2   0.002880         1.882365          0.000000      0.000000
```

Feedback

report_rc_corner_ratios

Reports the nets with the largest ratio of parasitics between corners.

Syntax

status *report_rc_corner_ratios*

```
-parasitic_corners corner1 corner2
[-type object]
[-nworst n1]
[-nbest n2]
```

Data Types

corner1	string
corner2	string
object	string
n1	integer
n2	integer

Arguments

`-parasitic_corners` *corner1 corner2*

Two corners for which to report the nets with the largest capacitance ratio.

`-type` *object*

Specifies the type of parasitic object to compare. Valid values are *ground_capacitor*, *coupling_capacitor*, *resistor*. The default is *ground_capacitor*.

`-nworst` *n1*

Specifies to report the largest ratios and the number of nets to report.

`-nbest` *n2*

Specifies to report the smallest ratios and the number of nets to report.

Description

This command reports the nets with the largest ratio of parasitics between corners.

If neither the *-nbest* or *-nworst* options is specified, the default is *-nworst 100*.

For each net, the ratio is the capacitance (or resistance) for corner 1 divided by the capacitance (or resistance) for corner 2.

This command is supported only for transistor-level GPDs.

Feedback

Examples

The following example shows a coupling capacitance report for corners typ and max, listing the 100 worst ratios.

```
pt_shell> report_rc_corner_ratios -parasitic_corners {typ max}\
            -type coupling_capacitor
****************************************
Report : RC Corner Ratios
Design : toprt
Version: Q-2019.12
Date   : Wed Nov 13 8:12:22 2019
****************************************

Parasitic Corner 1: typ
Parasitic Corner 2: max
Reporting 100 nworst nets with coupling_capacitor variation between
 corners

Net                      Ratio
===                      =====
sec_msb/cnt_blk1/n181    1.054146
min_msb/cnt_blk1/n224    1.052174
min_msb_led[3]           1.051689
min_msb/conv_blk1/n122   1.050078
```

The following example shows a coupling capacitance report, listing only the 10 best ratios.

```
pt_shell> report_rc_corner_ratios -parasitic_corners {typ max}\
            -type coupling_capacitor -nbest 10
****************************************
Report : RC Corner Ratios
Design : toprt
Version: Q-2019.12
Date   : Wed Nov 13 9:18:02 2019
****************************************

Parasitic Corner 1: typ
Parasitic Corner 2: max
Reporting 10 nbest nets with coupling_capacitor variation between corners

Net                      Ratio
===                      =====
min_msb/conv_blk1/n107   0.903774
min_msb/conv_blk1/n125   0.904806
min_lsb_led[0]           0.905907
min_lsb/conv_blk1/n89    0.906986
sec_msb/conv_blk1/n54    0.907177
min_lsb/cnt_blk1/n196    0.908030
min_lsb/conv_blk1/n97    0.908989
min_lsb_lef[4]           0.909031
min_msb/n128             0.909055
```

```
sec_msb/conv_blk1/n50    0.909223
-------------------------------------------------
```

report_redundant_memory_toggles

Analyzes input RTL FSDB for memory cells and reports redundant toggles.

Syntax

string *report_redundant_memory_toggles*

```
[-mem_state disabled_memory | enabled_write | disabled_write | all]
[cell_list]
```

Data Types

```
cell_list       list
```

Arguments

`-mem_state` *disabled_memory | enabled_write | disabled_write | all*

Use this option to choose the memory state to analyse and report redundant toggles.

`cell_list`

This command expects user to provide memory instance name for analysis. If no instance is specified all memory instances will be analysed.

Description

This command analyzes input RTL FSDB or VCD waveform and reports the redundant toggles for different memory states. The command uses input RTL FSDB and cycle power analysis to detect the time windows in which (a) Read-address/data, write-address/data inputs which are toggling despite disabled memory (b) Read-address inputs which are toggling despite enabled write-access (c) Write-address/data inputs which are toggling despite disabled write-access

The command dumps out an output FSDB file which shows the redundant time windows for each memory state. A value of 1 on the waveform signifies redundant toggles of the read/write data/address lines. Output FSDB also contains original input and output signals of the memory instances so user can validate and change RTL code.

The command reports the percentage of redundant toggles for each memory state.

User can use output FSDB and command output report to modify user's RTL and input FSDB for making architecture-level decisions and improving memory power.

Feedback

Examples

The following example shows the output report of *report_redundant_memory_toggles* command.

```
pwr_shell> report_redundant_memory_toggles
****************************************
Report : redundant_memory_toggles
      A: Read/Write addr/data toggling when memory is disabled
      B: Read addr toggling when write is enabled
      C: Write addr/data toggling when write is disabled
Design : wrapper_tcbram
Version: U-2022.12-SP3
Date   : Tue May 23 12:13:52
****************************************

Memory                          Memory                          Redundant
       Redundant           Redundant
Instance                        Cell                            Toggles (%)
       Toggles (%)         Toggles (%)
                                                                (A)
       (B)                 (C)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------
ram_inst/i_sram_tcbram          sram32x64_hscm4                  36.00
         45.00                 18.00
ram_inst1/i_sram_tcbram         sram32x64_hscm4                  36.00
         45.00                 18.00
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------
1
```

See Also

- report_memory_redundant_read_cycles
- report_memory_redundant_write_cycles
- set_memory_cell_info
- compute_metrics
- update_metrics

---

report_reference

Reports the references in current instance or design.

Feedback

Syntax

string *report_reference*

```
[-nosplit]
```

Arguments

`-nosplit`

Does not split lines if the column overflows.

Description

Displays information about all references in the current instance or current design. If the current instance is set, the report is generated relative to that instance; otherwise, the report is generated for the *current_design*.

Examples

This is an example of the *report_reference* output.

```
pt_shell> report_reference

****************************************
Report : reference
Design : middle
****************************************


Attributes:
    b - black box (unknown)
    h - hierarchical
    n - noncombinational

Reference       Library       Unit Area  Count  Total Area  Attributes
--------------------------------------------------------------------------
------
FD2               tech_lib         3.00     4          12.00      b
ND2               tech_lib         3.00     4          12.00      b
inter                              6.00     2          12.00      h
low                                2.00     2           4.00      h
--------------------------------------------------------------------------
------
Total 4 references                                     40.00
```

See Also

- report_hierarchy
- report_cell

Feedback

report_routed_nets

Reports the nets routed on specified layers and the total capacitance values of those nets.

Syntax

status *report_routed_nets*

`-layer` *layers*

Data Types

layers	list

Arguments

`-layer` *layers*

A list of one or more layers for which to report the routed nets. At least one layer is required. Glob-style wildcards (*) are supported.

Description

This command reports the nets routed on specified layers and the total capacitance values of those nets.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a routed net report. The report contains a section for each specified layer with a list of nets, the total capacitance for each net, and the percentage contribution of the layer to the total capacitance.

```
pt_shell> report_routed_nets {metal2 metal3}
==========================================
Total number of nets routed on metal2: 16
==========================================
Net Name   Total Capacitance   metal2 Capacitance   %metal2/Ct
========   =================   ==================   ==========
A0         0.092668            0.012534             13.525705
A1         0.092722            0.012547             13.531848
A2         0.092721            0.012547             13.531994
A3         0.092718            0.012544             13.529196
B0         0.089779            0.012808             14.266142
```

report_routing_rules

Reports non-default routing rules in the design.

Feedback

Syntax

status *report_routing_rules*

```
[-verbose]
[-nosplit]
[-significant_digits digits]
[-output file_name]
[-of_objects net_list]
[routing_rule_list]
```

Data Types

digits	int
file_name	string
net_list	list
routing_rule_list	string

Arguments

`-verbose`

Displays verbose non-default rule information.

`-nosplit`

Does not split lines if column overflows.

`-significant_digits digits`

Specifies the number of digits to the right of the decimal point that are to be reported. Allowed values are 0-13; the default is 2.

`-output file_name`

Generates a Tcl script that defines the specified routing rules. If the given *file_name* does not end with a .tcl file extension, one will be added. If the *routing_rule_list* option is used, the Tcl script will contain the non-default rule definitions. If *-of_objects net_list* option is used, the Tcl script will contain the net non-default rule assignments.

`-of_objects net_list`

Specifies the nets of which to report non-default rules.

`routing_rule_list`

Specifies the non-default routing rules to report. By default, the tool reports all non-default routing rules in the design.

Description

This command reports the details of the specified non-default routing rules, or the non-default routing rules assigned to the specified nets.

Feedback

Examples

The following example reports on the non-default rule named WideMetal.

```
prompt> report_routing_rules WideMetal
```

See Also

- set_routing_rule
- reset_routing_rule

---

report_rtl_metrics

Reports rtl cell characteristics.

Syntax

string *report_rtl_metrics*

```
[-view view_name]
[-hier_attributes attribute_list]
[-levels umber]
[-reg_attributes attribute_list]
[-mem_attributes attribute_list]
[-op_attributes attribute_list]
[-rtl_file_attributes attribute_list]
[-mod_attributes attribute_list]
[-sort_by single_attribute]
[-reverse_sort_by single_attribute]
[-nworst number]
[-by_hier hier_cell_list]
[-by_cell reg_cell_list]
[-by_mem mem_cell_list]
[-by_op operator_list]
[-by_file rtl_file_list]
[-nosplit]
[-disp_cell]
[-bus_merged_off]
[-report_nl_names]
[-disp_per_line]
[-csv csv_file]
[-filter expression]
[-tag]
[-list_attributes]
[-power_unit unit_type]
[-no_hier_name]
[-report_local_data]
[-report_vth_profile]
[object_list]
```

Feedback

Data Types

view_name	string
attribute_list	list
single_attribute	string
number	number
hier_cell_list	list
reg_cell_list	list
mem_cell_list	list
operator_list	list
rtl_file_list	list
csv_file	string
expression	string
object_list	list
unit_type	string

Arguments

`-view` *view_name*

With this option those views are supported: *hier*, *register*, *clock*, *memory*, *operator*, *rtl_file*.

The *-view hier* option is used to report various power exploration/ optimization attributes per hierarchy block. The supporting attributes are: total_power, total_leakage, total_switch, total_internal, total_sequential, sequential_leakage, sequential_switch, sequential_internal, total_combinational, combinational_leakage, combinational_switching, combinational_internal, total_memory, memory_leakage, memory_switching, memory_internal, sequential_annotation, combinational_annotation, memory_annotation, sequential_toggle, combinational_toggle, memory_toggle, glitch_toggle, average_cg, average_data_aware_cg, average_roade, average_dacgee, sequential_cells, combinational_cells, memory_cells, mbit_cells, icg_cells, gated_registers, inferred_gated_registers, instantiated_gated_registers, both_gated_registers, cell_area. The default display attributes are total_power, total_leakage, total_switching, total_internal, total_sequential, total_combinational, total_memory, average_cg, and cell_name.

The *-view register* option provides various information of registers in the design. The various attributes are: register_width, root_clk_name, root_clk_frequency, root_clk_toggle_rate, total_power, internal_power, leakage_power, switching_power, register_gated, clock_gating_type, clock_pin_toggle_rate, q_pin_toggle_rate, d_pin_toggle_rate, q_p_clk_toggle_rate, register_gating_effi, max_wns, max_tns, ROADE, DACGEE. The default display attributes are root_clk_name, root_clk_frequency, total_power, internal_power, leakage_power, switching_power, clock_gating_type, q_p_clk_toggle_rate, register_gating_effi.

The *-view clock* option is used to show various attributes of clock structure and clock gating cells. The clock structure table with column of clock name, frequency, total registers, total buffers, total inverters and maximum clock gate stage of the clock. The other is detailed clock gating cell table which contains root clock name, gating stage, gating element name, frequency, fanout number, register and clock gating element number, buffer and inverter number, intrinsic and incremental gating efficiency and clock gating efficiency.

The *-view memory* option is used to show various attributes of memory in the design. The various attributes are: cell_name, clock_name, clock_freq, total_power, internal_power, leakage_power, switching_power, ave_data_tr, ave_out_tr, ave_read_addr_tr, ave_write_addr_tr, read_condi_prob, write_condi_prob, read_condi_tr, write_condi_tr, read_condition, write_condition.

The *-view operator* option is used to show various attributes of operator in the design. The various attributes are: module_name, total_power, internal_power, leakage_power, switching_power, glitch_power, xref_location.

The *-view rtl_file* option is used to report various power attributes per rtl file and line. To view the report, you should first apply the *update_xref_file_metric* command. The supporting attributes are: total_power, internal_power, leakage_power, switching_power, glitch_power, total_power_sum, internal_power_sum, leakage_power_sum, switching_power_sum, glitch_power_sum, total_power_max, internal_power_max, leakage_power_max, switching_power_max, glitch_power_max.

`-hier_attributes` *`attribute_list`*

User can specify reporting column of hierarchical attributes table with *-view hier* option. If the user specifies *all*, then all the hierarchical attributes will be displayed.

`-levels` *`number`*

User can specify the maximum depth of the hierarchy in *-view hier* with the *-levels* option. Default is 1.

`-reg_attributes` *`attribute_list`*

User can specify reporting column of register attributes with *-view register* option. If the user specifies *all*, then all the register attributes will be displayed.

`-mem_attributes` *`attribute_list`*

User can specify reporting column of memory attributes with *-view memory* option. If the user specifies *all*, then all the memory attributes will be displayed.

`-op_attributes` *`attribute_list`*

User can specify reporting column of operator attributes with *-view operator* option. If the user specifies *all*, then all the memory attributes will be displayed.

`-rtl_file_attributes` *`attribute_list`*

User can specify reporting column of rtl file attributes with *-view rtl_file* option. If the user specifies *all*, then all the rtl file attributes will be displayed.

`-mod_attributes` *`attribute_list`*

User can specify reporting column of module view attributes with *-view mod_cg_savings* option. If the user specifies *all*, then all the mod view attributes will be displayed.

`-sort_by` *`single_attribute`*

Specifies an attribute for sorting option in each view option.

`-reverse_sort_by` *`single_attribute`*

Specifies an attribute for reverse sorting option in each view option.

`-nworst` *`number`*

Specifies *number* of displayed element which is sorted by the *-sort_by* option.

`-by_hier` *`hier_cell_list`*

With the *-view register* option, this option specifies results of register cells which are to be aggregated from the specified hierarchy blocks.

`-by_cell` *`reg_cell_list`*

With the *-view register* option, this option specifies results of register cells which are specified with the given cell list.

`-by_mem` *`mem_cell_list`*

With the *-view memory* option, this option specifies results of memory cells which are specified with the given cell list.

`-by_op` *`operator_list`*

With the *-view operator* option, this option specifies results of operators which are specified with the given operator_list.

`-by_file` *`rtl_file_list`*

With the *-view rtl_file* option, this option displays results of rtl files which are specified with the given rtl_file_list.

`-nosplit`

This option specifies that report lines should not be split when names are longer than the space provided in the report. The option can make it easier for scripts to parse the result, but might make it harder for human to read the columns.

`-disp_cell`

This option gives detailed cell list report for the *-view operator* option: This option gives operator name, cell counts, each cell's full_name under the operator.

`-bus_merged_off`

With the *-view register* option, this option turn off the bus name merged option.

`-report_nl_names`

With the *-view register* option, this option prints the gate level names of the register in the report. By default the RTL names of the register gets printed in the report.

`-disp_per_line`

This option displays per line report for the *-view rtl_file* option:

`-csv` *`csv_file`*

This option specifies that the report is written out in a comma-separated value format. The *csv_file* specifies the csv file name for the report.

`-filter` *`expression`*

Filters the collection with the *expression* value. For objects the expression is evaluated based on the report columns values. If the expression evaluates to true, the object is included in the report.

`-tag`

This option shows Tag names for each columns of any view mentioned. This option is supported for all views.

`-list_attributes`

This option prints all the attribute names for each columns of all the views, along with the detailed description. This option is supported with the filter option (for view and attribute)

`-power_unit` *`unit_type`*

This option helps to display power numbers in unit types like W(Watts), mW(milliWatt), uW(microWatt), nW(nano Watt) This option is supported for all views.

`-no_hier_name`

This option prints only the register names i.e without their hierarchical path. This option is only supported for Hier View.

`-report_local_data`

This option adds an extra row after each hierarchy reporting the attributes for the leaf cells instantiated in that hierarchy without adding the aggregates from the lower hierarchies. The rows with local data can be identified by the string _local_ appended after the hierarchical path. This option is only supported for Hier View with the application variable power_enable_new_rrm_view set to true.

`-report_vth_profile`

This option adds extra columns for threshold voltage groups at the end of hierarchy view report and prints the number of leaf cells per threshold voltage group present in the hierarchy. The number of cells is counted based on the threshold voltage defined for the corresponding library cells. If the threshold voltage group is not defined on the library cell, the cell is assigned to the default threshold voltage group. Only those leaf cells are considered which contribute to the total power calculated for the hierarchy. Others like logic constant cells and cells without supply nets are ignored. The threshold voltage group names cannot be used as attributes for filtering or sorting the report. This option is only supported for Hier View with the application variable power_enable_new_rrm_view set to true.

Description

This command provides various information in different views to analyze user's RTL for making architecture-level decisions. This information spans from existing design characteristics to tool inferred attributes that will help the user make design decisions.

Mnemonics for NA type in report:

NA_CP0 - Root clock toggle rate is zero

NA_GCP0 - Gated register clock pin toggle rate is zero

NA_Q0 - Gated register Q pin toggle rate is zero

NA_Q>CP - Gated register Q pin toggle rate is greater than clock pin

NA_GC>C - Gated register clock pin toggle rate is greater than root clock toggle rate

NA_NOREG - No registers in the hierarchy

NA - Any other reason

To use the command, you should first apply the *update_power* and *update_metrics* command or apply the *compute_metrics* command.

Feedback

Examples

The following example shows the default report of *-view hier*:

```
pwr_shell> report_rtl_metrics -view hier

****************************************
Report : report_rtl_metrics
      -view hier
Design : top
****************************************


total_power   total_leakage   total_switching   total_internal
 total_sequential   total_combinational   total_memory   average_cg
 cell_name
-----------   -------------   ---------------   --------------
 ----------------   -------------------   ------------   ----------
 -----------------------------------
6.72e-05      1.69e-08        1.40e-05          5.32e-05         4.47e-05
           2.25e-05              0.00e+00        50.79       top
1.79e-05      1.05e-08        1.36e-06          1.65e-05
 1.74e-05           4.83e-07              0.00e+00        71.93
 top/flat_inst

To highligt the specific hiearachy module, user can put hier cell list as
 follows:

pt_shell> report_rtl_metrics -view hier {BM*}
****************************************
Report : report_rtl_metrics
      -view hier
Design : top
****************************************


total_power   total_leakage   total_switching   total_internal
 total_sequential   total_combinational   total_memory   average_cg
 cell_name
-----------   -------------   ---------------   --------------
 ----------------   -------------------   ------------   ----------
 -----------------------------------
5.58e-05      1.44e-06        1.10e-05          4.34e-05         3.12e-05
           2.46e-05              0.00e+00        40.27       top/BM0

The following example shows the report of -view hier with
 -hier_attributes:

pt_shell> report_rtl_metrics -view hier -hier_attributes {cell_area
 total_power}
****************************************
Report : report_rtl_metrics
      -view hier
```

```
Design : top
****************************************


total_power   cell_area   cell_name
-----------   ---------   ----------------------------------
6.72e-05      1.29e+03    top
1.79e-05      7.26e+02    top/flat_inst
```

The following example shows the default report of **-view register**:

```
pt_shell> report_rtl_metrics -view register
****************************************
Report : report_rtl_metrics
      -view register
Design : top
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
Register                              Root Clk       Root Clk       Total
       Internal       Leakage          Switching      Clock          Q/Clk
         Register
Name                                  Name           Freqency       Power
       Power          Power            Power          Gating         Toggle
         Gating
                                                      (MHz)
                                                         Type           Rate
 (%)       Effi. (%)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
B_reg[7:4,3:0]                            clk            1000.0
 6.368e-06      6.139e-06      2.532e-08      2.037e-07      Instantiated
   14.7           49.9
C_reg[7:4]                                clk            1000.0
 2.279e-06      2.266e-06      1.266e-08      0              Instantiated
   7.4            49.9
C_reg[3:0]                                clk            1000.0
 7.039e-07      6.913e-07      1.266e-08      0              Instantiated
   13.5           90.2
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
```

B_reg[7:4] and B_reg[3:0] shares same ICG cell. Thus, the command shows bus merged name and related report as above.
To turn off the bus merged option, we provide -bus_merged_off option.
The following example shows the report of **-view register** with **-bus_merged_off**:

Feedback

```
pt_shell> report_rtl_metrics -view register -bus_merged_off
****************************************
Report : report_rtl_metrics
      -view register
Design : top
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
Register                              Root Clk       Root Clk       Total
       Internal       Leakage          Switching      Clock          Q/Clk
         Register
Name                                  Name           Freqency       Power
       Power          Power            Power          Gating         Toggle
         Gating
                                                     (MHz)
                                                        Type           Rate
 (%)       Effi. (%)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
B_reg[7:4]                                clk            1000.0
 3.185e-06      3.07e-06       1.266e-08      1.018e-07      Instantiated
   14.7           49.9
B_reg[3:0]                                clk            1000.0
 3.183e-06      3.069e-06      1.266e-08      1.019e-07      Instantiated
   14.6           49.9
C_reg[7:4]                                clk            1000.0
 2.279e-06      2.266e-06      1.266e-08      0              Instantiated
   7.4            49.9
C_reg[3:0]                                clk            1000.0
 7.039e-07      6.913e-07      1.266e-08      0              Instantiated
   13.5           90.2
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
```

The following example shows a usage of the **-sort_by** option with **-reg_attributes** option:

```
pt_shell> report_rtl_metrics -view register -reg_attributes
 {q_p_clk_toggle_rate} -sort_by q_p_clk_toggle_rate
****************************************
Report : report_rtl_metrics
      -view register
Design : top
Version: R-2020.09-DEV-20200720
Date   : Tue Jul 21 16:55:39 2020
****************************************

------------------------------------------------
```

```
Register                                Q/Clk
Name                                    Toggle
                                        Rate (%)
--------------------------------------------------
BM0/B_12345678901234567890123456789012345_reg[7:4,3:0]
                                        14.7
BM0/C_12345678901234567890123456789012345_reg[3:0]
                                        13.4
A_reg[3:2]                              8.1
BM0/C_12345678901234567890123456789012345_reg[7:4]
                                        7.6
BM0/A_reg[2:1,0]                        4.8
A_reg[1:0]                              4.7
--------------------------------------------------
1
```

The following example shows the usage of **-by_hier** option. The report shows registers under the specific hierarchy module.

```
pt_shell> report_rtl_metrics -view register -by_hier { BM* }
 -reg_attributes {total_power}
****************************************
Report : report_rtl_metrics
      -view register
Design : top
****************************************

--------------------------------------------------
Register                                Total
Name                                    Power

--------------------------------------------------
BM0/B_12345678901234567890123456789012345_reg[7:4,3:0]
                                        6.301e-06
BM0/A_reg[2:1,0]                        4.174e-06
BM0/C_12345678901234567890123456789012345_reg[7:4]
                                        2.269e-06
BM0/C_12345678901234567890123456789012345_reg[3:0]
                                        7.152e-07
--------------------------------------------------
```

To view specific register or group of register report, we provide **-by_cell** option.
The following example shows the usage of **-by_cell** option:

```
pt_shell> report_rtl_metrics -view register -by_cell {A_reg*}
 -reg_attributes {total_power}
****************************************
Report : report_rtl_metrics
      -view register
Design : top
```

```
****************************************

--------------------------------------------------
Register                              Total
Name                                  Power

--------------------------------------------------
A_reg[1:0]                            2.699e-06
A_reg[3:2]                            1.459e-06
--------------------------------------------------
```

The following example shows the default report of **-view clock**:

```
pt_shell> report_rtl_metrics -view clock
****************************************
Report : report_rtl_metrics
      -view clock
Design : top
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------

 Clock Structure Summary
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------
Clock          Frequency      # of total     # of Direct    # of total
 # of total     # of total     # of total     # of Direct    # of Direct
   Max. # of
               (MHz)          Registers      Conn. Regs     Gated Regs
 Clock Gates    Buffers        Inverters      Conn. Bufs     Conn. Ivts
   C.G. stage
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------
clk_123456789012345_clk
               1000.0         7              3              4
 3               3              0              1              0
   2
clk            500.0          1              1              0
 0               0              0              0              0
   0
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------

 Clock Gate Structure Details
```

Feedback

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------
Root Clk      Clock          Gating                    Frequency      #
 of           # of           # of           # of           # of
 Intrinsic      Incremental    Clock Gating
Name          Gating Stage   Element                   (MHz)          Fan
 Out        Clock Gates    Registers      Buffers        Inverters
 C.G. Eff.      C.G. Eff.      Efficiency
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------
clk_123456789012345_clk
              1              BM0/icg2                  494.5          3
             0              3              1              0
 50.5%          50.5%          21.662%
clk_123456789012345_clk
              2              BM0/icg1                  252.2          1
             1              0              1              0
 74.8%          74.8%          10.683%
clk_123456789012345_clk
              1              BM0/icg3                  100.4          1
             0              1              0              0
 60.2%          15.2%           2.168%
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------



pwr_shell> report_rtl_metrics -view hier -filter "average_cg > 0 &&
 average_cg < 50"
****************************************
Report : report_rtl_metrics
      -view hier
      -filter average_cg > 0 && average_cg < 50
Design : ethmac
****************************************

total_power   total_leakage   total_switching   total_internal
 total_sequential   total_combinational   total_memory   average_cg
 cell_name
-----------   -------------   ---------------   --------------
 ---------------   -------------------   ------------   ----------
 ----------------------------------
2.46e-06      2.16e-06        2.36e-08          2.79e-07
 1.95e-06          5.10e-07              0.00e+00         28.4
 ethmac/rxethmac1
2.23e-06      1.78e-06        4.97e-08          3.93e-07
 1.70e-06          5.22e-07              0.00e+00         14.9
 ethmac/txethmac1
```

Feedback

```
1.89e-06      1.72e-06        1.13e-08          1.59e-07
 1.48e-06           4.09e-07              0.00e+00         37.4
 ethmac/maccontrol1
7.06e-07      6.43e-07        1.54e-10          6.31e-08
 6.02e-07           1.04e-07              0.00e+00         21.4
 ethmac/macstatus1
1


pwr_shell> report_rtl_metrics -view register -filter
 "clock_gating_type==Inferred && register_gating_effi > 50"
****************************************
Report : report_rtl_metrics
      -view register
      -filter clock_gating_type==Inferred && register_gating_effi > 50

Design : ethmac
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
Register                              Root Clk       Root Clk       Total
       Internal       Leakage          Switching      Clock          Q/Clk
         Register
Name                                  Name           Freqency       Power
       Power          Power            Power          Gating         Toggle
         Gating
                                                     (MHz)
                                                        Type           Rate
 (%)       Effi. (%)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
miim1/clkgen/Counter_reg              ethmac_wb_clk_i
                                                        100.0
 5.614e-08      4.44e-08       5.066e-09      6.668e-09      Inferred
   69.1           74.2
miim1/clkgen/Mdc_reg                  ethmac_wb_clk_i
                                                        100.0
 2.893e-08      2.166e-08      6.103e-09      1.16e-09       Inferred
   7.9            74.2
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
1

pwr_shell> report_rtl_metrics -view hier -filter "average_cg < 50 &&
 total_power < 0.00001 && total_leakage < 0.000001"
****************************************
Report : report_rtl_metrics
      -view hier
```

```
      -filter average_cg < 50 && total_power < 0.00001 && total_leakage <
 0.000001
Design : ethmac
****************************************

total_power   total_leakage   total_switching   total_internal
 total_sequential   total_combinational   total_memory   average_cg
 cell_name
-----------   -------------   ---------------   --------------
 ----------------   --------------------   ------------   ----------
 -----------------------------------
7.06e-07      6.43e-07        1.54e-10          6.31e-08
 6.02e-07           1.04e-07              0.00e+00         21.4
 ethmac/macstatus1
1


pwr_shell> report_rtl_metrics -view register -filter "data_aware_cg_effi
 > 50 && q_p_clk_toggle_rate > 50" -reg_attributes {data_aware_cg_effi
 q_p_clk_toggle_rate}
****************************************
Report : report_rtl_metrics
      -view register
      -filter data_aware_cg_effi > 50 && q_p_clk_toggle_rate > 50
Design : ethmac
****************************************

------------------------------------------------------------------
Register                          Q/Clk          Data Aware
Name                              Toggle         Clock Gating
                                  Rate (%)       Effi. (%)
------------------------------------------------------------------
miim1/clkgen/Counter_reg[0:1]     73.4           73.4
txethmac1/random1/x_reg[0:9]      53.7           53.7
miim1/clkgen/Counter_reg[3]/Q,
miim1/clkgen/Counter_reg[2]/Q
                                  69.1           92.0
------------------------------------------------------------------
1

pwr_shell> report_rtl_metrics -view rtl_file -sort_by total_power_sum
 -nworst 3
****************************************
Report : report_rtl_metrics
      -view rtl_file
      -sort_by total_power_sum
      -nworst 3
Design : ethmac
****************************************

------------------------------------------------------------------------
------------------------------------------------------------------------
-----------------------------
```

```
RTL                      Total          Total          Internal
 Internal       Switching      Switching      Leakage        Leakage
   Glitch         Glitch
File                     Power          Power          Power
 Power          Power          Power          Power          Power
   Power          Power
Name                     Sum            Max            Sum            Max
            Sum            Max            Sum            Max
 Sum            Max
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------
/global/gts_corpac3/dkumari/ethmac/INPUT/rtl/verilog/eth_wishbone.v
                         9.187e-06      1.427e-06      6.374e-06
 1.148e-06      5.767e-07      3.556e-07      2.235e-06      1.942e-07
   8.91e-09       7.87e-09
/global/gts_corpac3/dkumari/ethmac/INPUT/rtl/verilog/eth_register.v
                         3.028e-06      2.11e-06       1.586e-06
 1.174e-06      2.483e-07      2.401e-07      1.194e-06      9.285e-07
   0              0
/global/gts_corpac3/dkumari/ethmac/INPUT/rtl/verilog/ethmac.v
                         1.656e-06      1.075e-06      1.388e-06
 9.695e-07      3.239e-08      9.774e-09      2.36e-07       1.03e-07
   0              0
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------
1

pwr_shell> report_rtl_metrics -view rtl_file -disp_per_line -by_file
 {/global/gts_corpac3/dkumari/ethmac/INPUT/rtl/verilog/ethmac.v} -sort_by
 total_power -nworst 3
**************************************
Report : report_rtl_metrics
      -view rtl_file
      -sort_by total_power
      -nworst 3
      -disp_per_line
Design : ethmac
Version: S-2021.06-SP5-DEV-20220308
Date   : Wed Mar  9 22:03:35 2022
**************************************

-------------------------------------------------------------------------
-----------------------------------------
RTL                      Line           Total          Internal
 Switching      Leakage        Glitch
File                     Number         Power          Power
 Power          Power          Power
Name
-------------------------------------------------------------------------
-----------------------------------------
/global/gts_corpac3/dkumari/ethmac/INPUT/rtl/verilog/ethmac.v
```

```
                          549             1.075e-06      9.695e-07
 2.862e-09      1.03e-07       0
/global/gts_corpac3/dkumari/ethmac/INPUT/rtl/verilog/ethmac.v
                          985             1.155e-07      1.092e-07      0
            6.341e-09      0
/global/gts_corpac3/dkumari/ethmac/INPUT/rtl/verilog/ethmac.v
                          912             7.999e-08      7.617e-08      0
            3.822e-09      0
--------------------------------------------------------------------------
----------------------------------------
1
```

See Also

- compute_metrics
- update_power
- update_metrics

---

report_scale_parasitics

Reports the parasitic resistance and capacitance scaling factors previously set by the *scale_parasitics* command.

Syntax

status *report_scale_parasitics*

`[net_list]`

Data Types

net_list list

Arguments

net_list

Limits the reporting of net-specific scaling settings to only the listed nets. By default, the command reports all net-specific scaling settings, in addition to the global scaling settings.

Description

The *report_scale_parasitics* command reports the parasitic resistance and capacitance scaling factors previously set by the *scale_parasitics* command. Both the global and net-specific scaling settings are reported.

Examples

The following example reports the parasitic scaling factors.

Feedback

```
pt_shell> report_scale_parasitics
Global factors:
Cap factor      : 1.400000
Res factor      : 1.500000
Coupling factor : 1.350000
Net                    Cap factor   Res factor   Coupling factor
-------------------------------------------------------------------
net_sdram_DQ_in[0]     1.200000     1.500000     1.200000
net_sdram_DQ_in[1]     1.200000     1.500000     1.200000
net_sdram_DQ_in[2]     1.200000     1.500000     1.200000
net_sdram_DQ_in[3]     1.200000     1.500000     1.200000
1
```

See Also

- read_parasitics
- reset_scale_parasitics
- scale_parasitics

---

report_sense

Reports user-specified unatenes and sense propagation constraints for data or clock.

Syntax

string *report_sense*

```
[-type clock | data]
[-clocks clock_list]
[-nosplit]
object_list
```

Data Types

```
clock_list       list
object_list      list
```

Arguments

`-type clock | data`

Specifies whether the type of sense applies to clock or data networks. The allowed values are *clock* (the default) and *data*.

`-clocks clock_list`

Reports the specified clock objects. If you do not use the *-clocks* option, the tool reports all clocks passing through the given pin or arc objects.

-nosplit

Prevents line splitting if a column overflows.

object_list

Lists of ports, pins, or cell timing arcs to report.

Description

This command reports the user-defined constraints applied to the senses and unateness propagation of data and clock signals. This reports are distinguished based on whether the sense narrows down the clock waveform, or the sense creates a new clock waveform, as specified by the attributes G for source latency waveform and P for primary clock waveform.

Examples

The following example specifies and reports a stop sense propagation through an arc:

```
pt_shell> set_sense -stop_propagation [get_timing_arcs -from u2/A -to
 u2/Z] \\
          -clock [get_clock clk12]
pt_shell> update_timing
pt_shell> report_sense -type clock

****************************************
Report : sense
         -type clock
Design : simple_2ff
 ...
****************************************


Attributes:
    G - Generated
    P - Primary
Object                                  Clock         Sense
 Attr
--------------------------------------------------------------------------
---
u2/A -> Z                               clk12         stop_prop
1
```

The following example selects positive unateness for pin MUX1/Z for all clocks:

```
pt_shell> set_sense -positive -clock clk MUX1/Z
pt_shell> update_timing
pt_shell> report_sense
****************************************
Report : sense
         -type clock
Design : nonunate
```

```
 ...
****************************************

Object                    Clock               Sense
------------------------------------------------------------
MUX1/Z                      *                 positive
```

See Also

- remove_sense
- set_sense

---

report_sensitivity_lib_mapping

Reports the mapping between base library and sensitivity augmented library.

Syntax

string *report_sensitivity_lib_mapping*

```
[libraries]
```

Data Types

```
libraries                 list
```

Arguments

libraries

Specifies a list of libraries to report mapping of base library and sensitivity agumented library. If this option is omitted, the report covers all of libraries in the current design.

Description

The *report_sensitivity_lib_mapping* command reports mapping between base library and sensitivity augmented library, which is defined in *define_sensitivity_lib_mapping*. *define_sensitivity_lib_mapping* is required in PrimeShield Project Sicily.

Examples

The following script enables Project Sicily Spice2Design flow, defines mapping between base library and sensitivity side-file, then report the mapping.

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_spice2design_analysis true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]
```

Feedback

```
pt_shell> report_sensitivity_lib_mapping orig
  Lib              side_file
================================================
  orig             side.db
================================================
1
```

See Also

- define_sensitivity_lib_mapping
- reset_sensitivity_lib_mapping
- ps_enable_spice2design_analysis
- ps_enable_pvt_explorer_analysis

---

report_sensitivity_power_lib_mapping

Sets the options for sensitivity library check for PrimePower.

Syntax

integer *report_sensitivity_power_lib_mapping*

```
[-output_file_name]
[-dump_sensitivity_leak_tables]
```

Data Types

dump_sensitivity_leak_tables	boolean
output_file_name	string

Arguments

`-dump_sensitivity_leak_tables`

Specifies that all the content of the sensitvity library file needs to be dumped into either the default file i.e *report_s2s_power_content.rpt* or the file specified by output_file_name *argument .*

`-output_file_name`

Specifies the file in which the power sensitivity warnigs is to be written. The default is *report_s2s_power_content.rpt*.

Description

This command runs the following consistency check right after the libraries are loaded and dump the warnings in a file. 1. Warning: 1-1 mapping of leakage power table a. When condition b. Related_pg_pin 2. Warning: 1-1 mapping of rise and fall power table a. When

condition and related_pg_pin are same b. Number of input indices (input slew, output load) are same in nominal and side file 3. Value check (for tables that are a structural match) (Inputs are main library and side-file sensitivity library) a. Range of input indices are same in nominal and side file b. Warning if NLPM nominal value in side-file has larger than 1pJ/1% gap v.s. main library The command should be enabled before *update_power*.

Examples

The following example runs report_sensitivity_power_lib_mapping and logs all the warning in *sample_report.rpt*.

```
pt_shell> report_sensitivity_power_lib_mapping -output_file_name
 sample_report.rpt
Information: Detailed sensitivity report generated successfully in
 sample_report.rpt file

Below is a snippet of the report with the warnings.
Warning: Nominal Values = 0.00100495 is not matching S2S Values = 1234
Warning: Nominal Values = 0.000527413 is not matching S2S Values =
 0.000488281
```

The following example runs report_sensitivity_power_lib_mapping and dumps all the content of side_file. in *report_s2s_power_content.rpt*.

```
pt_shell> report_sensitivity_power_lib_mapping
 -dump_sensitivity_leak_tables
Information: Detailed sensitivity report generated successfully in
 report_s2s_power_content.rpt file

Below is a snippet of the report when sensitivity library file content is
 dumped.
Internal power ()
  Related_pg_pin: VDD
  When: CP&!D&!QN
       VARIABLE_1: input_transition_time
       INDEX_1:  0.005 0.0207 0.0521 0.1149 0.2406 0.4919 0.9946     2

       fall_power

       VALUES :
              0.0005  0.0002  0.0001  0.0003  0.0002  0.0002  0.0002
 0.0002
       VARIABLE_1: input_transition_time
       INDEX_1:  0.005 0.0207 0.0521 0.1149 0.2406 0.4919 0.9946     2

      rise_power

       VALUES :
              0.0003 -0.0000 -0.0001 -0.0001 -0.0002 -0.0002 -0.0002
 -0.0002
Internal power sensitivity table:
  Related_pg_pin: VDD
```

```
  When: CP&!D&!QN
  Param_name: ids0multp_mos_svt
  Pert_ratio: -0.100000
  Table_type: rise
  VARIABLE_1: input_transition_time
      INDEX_1:  0.00500000  0.02070000
      VALUES:  0.00002198  0.00000758

Leakage power ():
  Related_pg_pin: VDD
  When: !CP !D !QN !SDN
  value: 1.234
Leakage power sensitivity:
  Param_name: ddvtn_mos_svt
  Pert_value: -0.010000
  Leakage_power_sensitivity: -2.31625e-10
```

report_si_aggressor_exclusion

Reports the exclusive groups set by the *set_si_aggressor_exclusion* command.

Syntax

status *report_si_aggressor_exclusion*

```
[-rise]
[-fall]
[-nosplit]
[net_list]
```

Data Types

net_list	list

Arguments

`-rise`

Reports all the exclusive groups that have been set as exclusive in the rise direction. If neither the *-rise* nor the *-fall* options are specified, the exclusive groups of the aggressor nets *anets* in both rise and fall directions are reported.

`-fall`

Reports all the exclusive groups that have been set as exclusive in the fall direction. If neither the *-rise* nor the *-fall* options are specified, the exclusive groups of the aggressor nets *anets* in both rise and fall directions are reported.

`-nosplit`

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information

is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

net_list

Specifies the list of nets for exclusive groups you want reported. You can use this option with either the *-rise* or *-fall* options to specify exclusive groups only in that particular direction.

Description

The *report_si_aggressor_exclusion* command reports all the exclusive groups that have been set by the *set_si_aggressor_exclusion* command.

Examples

The following example shows how the exclusion on nets *SCAN_LOGIC** for rise direction is reported.

```
pt_shell> set_si_aggressor_exclusion [get_nets SCAN_LOGIC*] -rise
1
pt_shell> report_si_aggressor_exclusion
****************************************
  Report : report_si_aggressor_exclusion
           -rise
  Design : top
  Version: X-2005.12-DEV
  Date   : Thu Aug 11 14:43:46 2005
  ****************************************

  Attributes:
    R - Exclusive for Rise direction
    F - Exclusive for Fall direction

  Exclusive group         Exclusive type      Number of Active Aggressors
  ------------------------------------------------------------------------
  SCAN_LOGIC1 SCAN_LOGIC2  R                  1
1
```

The following example shows how the exclusive groups on specific nets can be reported for rise or fall direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets {Agg1 Agg2 Agg3}]
-number_of_active_aggressors 2 -rise
1
pt_shell> set_si_aggressor_exclusion [get_nets {Agg2 Agg4}]
-number_of_active_aggressors 1 -fall
1
pt_shell> report_si_aggressor_exclusion [get_nets Agg2] -rise
****************************************
  Report : report_si_aggressor_exclusion
           -rise
```

```
          -fall
  Design : top
  ...
  ****************************************

  Attributes:
    R - Exclusive for Rise direction
    F - Exclusive for Fall direction

  Exclusive group         Exclusive type      Number of Active Aggressors
  ---------------------------------------------------------------------------
  Agg1 Agg2 Agg3          R                   2
1

pt_shell> report_si_aggressor_exclusion [get_nets Agg2] -fall
****************************************
  Report : report_si_aggressor_exclusion
           -rise
           -fall
  Design : top
  ...
  ****************************************

  Attributes:
    R - Exclusive for Rise direction
    F - Exclusive for Fall direction

  Exclusive group         Exclusive type      Number of Active Aggressors
  ---------------------------------------------------------------------------
  Agg2 Agg4               F                   1
1
```

See Also

- remove_si_aggressor_exclusion
- report_delay_calculation
- report_noise_calculation
- set_si_aggressor_exclusion
- set_si_delay_analysis
- set_si_noise_analysis
- si_analysis_logical_correlation_mode

report_si_bottleneck

Reports the nets that have the largest crosstalk effects that contribute to timing violations.

Feedback

Syntax

status *report_si_bottleneck*

```
[-cost_type type]
[-slack_lesser_than slack]
[-minimum_active_aggressors active_aggressor_count]
[-delta_delay_threshold threshold]
[-min]
[-max]
[-all_nets]
[-nets list]
[-max_nets number]
[-include_clock_nets]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-significant_digits digits]
[-nosplit]
[-sms_scenarios sms_scenarios_list]
```

Data Types

type	float
slack	float
active_aggressor_count	int
threshold	float
list	list
number	int
pre_command_string	string
post_command_string	string
digits	int
sms_scenarios_list	collection

Arguments

`-cost_type` *type*

Sorts nets based on the specified cost types. The following cost types return a list of victim nets:

- *delta_delay* - The worst delta delay on the victim net. This is useful for finding the largest delta delays in the design that contribute to failing paths.
- *delta_delay_ratio* - The worst delta delay ratio (delta delay divided by total stage delay) on the victim net. This is useful for finding the delta delays that result in the largest percentage increase of a stage delay in failing paths.
- *total_victim_delay_bump* - The height of the largest crosstalk bump on the victim net. This is useful for finding the strongest electrical couplings in the design that contribute to failing paths.

The following cost type returns a list of aggressor nets:

- *delay_bump_per_aggressor* - The aggressor net with cross-coupled, slack-qualifying victim nets with the largest sum of bump voltages on those victims induced by the aggressor. This is useful for finding aggressors that result in the largest amount of electrical coupling to victim nets in failing paths.

`-slack_lesser_than` *`slack`*

Applies the cost function only to victim nets with slack less than the specified limit. The default is zero. Specify a positive value to find more bottleneck nets, and vice versa.

`-minimum_active_aggressors` *`active_aggressor_count`*

Specifies the minimum number of active aggressors for the net to be selected. The default is 1.

`-delta_delay_threshold` *`threshold`*

Reports only the victim nets whose delta delay cost exceeds the specified threshold. The threshold must be at least zero. The default is zero. This option applies only if the *-cost_type* option is set to *delta_delay*.

`-min`

Considers only minimum-delay (hold) constraints.

`-max`

Considers only maximum-delay (setup) constraints.

By default, the command considers both min and max constraints.

`-all_nets`

Reports all nets with SI cost that contribute to negative path slacks. By default, the command reports the 20 nets with the worst bottleneck cost.

`-nets` *`list`*

Restricts the bottleneck analysis to only the nets specified in a list or collection of nets. By default, the command considers all nets in the design.

`-max_nets` *`number`*

Specifies the maximum number of nets to report. The default is 20.

`-include_clock_nets`

Includes the clock nets in bottleneck analysis. By default, clock nets are not included.

`-pre_commands` *`pre_command_string`*

Specifies a string of commands to be executed in the worker context before execution of the merged reporting command.

`-post_commands` *`post_command_string`*

Specifies a string of commands to be executed in the worker context after execution of the merged reporting commands.

The *-pre_commands* and *-post_commands* options are valid only for distributed multi-scenario (DMSA) analysis. Commands are grouped using the ";" character. The maximum size of a command is 1000 characters.

`-significant_digits` *`digits`*

Specifies the number of digits to display; the allowed range is 0 to 13. The default is set by the *report_default_significant_digits* variable.

`-nosplit`

Prevents splitting of lines in the report when a column overflows.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only SI bottleneck information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *report_si_bottleneck* command reports the nets that have the largest crosstalk effects that contribute to timing violations. You can choose any one of the following "cost functions" to consider in the analysis:

- *delta_delay* (the default) -- Reports the victim nets with the worst change in absolute delay
- *delta_delay_ratio* -- Reports the victim nets with the worst change in delay as a fraction of the total stage delay
- *total_victim_delay_bump* -- Reports the victim nets with the largest crosstalk bumps
- *delay_bump_per_aggressor* -- Reports the aggressor nets that produce the largest total bump voltages in victim nets of failing paths

By default, the analysis considers only crosstalk effects that contribute to timing violations. Paths with zero or better slack are ignored. You can change this by using the *-slack_lesser_than option*.

By default, the analysis considers both maximum-delay (setup) and minimum-delay (hold) constraints. To restrict the analysis to only one constraint type, use the *-max* or *-min* option.

To create a collection of victim or aggressor nets reported by the *report_si_bottleneck* command, use the *get_si_bottleneck_nets* command.

If a victim fails to meet the slack criteria only for one of its transition senses (rise or fall), only the timing information for the failing sense is considered.

By default, clock nets are not included in bottleneck victim analysis, as the slack of the clock nets is infinite. However, clock nets can be included as victims in the bottleneck search by using the *-include_clock_nets* option. Even without this option, clock aggressor nets are returned by the *total_victim_delay_bump* cost type.

Crosstalk problems can be caused by one or more of the following factors:

- Large coupling capacitance
- Weak victim net driver or slow victim transition
- Strong aggressor net driver

Depending on the relative victim and aggressor slacks, one of the following repairs could be considered:

- Downsize the aggressor net driver to a weaker driver
- Resize the victim net driver to a stronger driver
- Specify physical decoupling with the *set_coupling_separation* command
- Insert a buffer on the victim net to break up long coupled routes

When sizing cells, it can be useful to use the *get_alternative_lib_cells* command to report equivalent cells, or the *report_alternative_lib_cells* command to evaluate their potential slack improvement.

The aggressor-based cost factor can be useful for finding strong aggressors that attack many failing victim nets. If the aggressor has positive setup slack, it might be preferable to downsize the aggressor driver and weaken its effect over multiple victims, or to separate it from the victims using the *set_coupling_separation* command. When performing such an aggressor downsizing, ensure that the resulting weaker aggressor driver does not violate any library max_capacitance or max_transition DRC requirements.

When many aggressors attack a victim net, you can take advantage of the statistical nature of many aggressors where the chances of all of them switching at the same time are very low. To find the failing victim nets that are attacked by many aggressors, you can use the *-minimum_active_aggressors* option. The number of active aggressors is

defined as the number of effective aggressors that are active for the worst-case aggressor alignment.

For example, a victim net can have ten effective aggressors. However, for any given min/max rise/fall scenario, perhaps no more than five of these ten effective aggressors are active in the worst-case alignment for that sense. Such a victim would not meet a *-minimum_active_aggressors* value higher than five.

To exclude specific crosstalk victim/aggressor interactions from the analysis without changing the parasitics, use the *set_si_delay_analysis -exclude* command.

For bottleneck analysis, crosstalk analysis must be enabled (*si_enable_analysis* set to *true*). Bottleneck analysis automatically sets the *timing_save_pin_arrival_and_slack* variable to *true*.

Examples

The following examples show SI bottleneck reports using the default cost type (*delta_delay*) and alternative cost types.

```
pt_shell> report_si_bottleneck
****************************************
Report : si_bottleneck
        -cost_type delta_delay
        -slack_lesser_than 0
        -max_nets 20
        -significant_digits 4
        -minimum_active_aggressors 1
        -min
        -max
    ...
****************************************

 Bottleneck Cost:  delta_delay
 net                                                  cost
 -----------------------------------------------------------
 DX23[17]                                              5.5762
 REG_ADDR[2]                                           5.0746
 REG_DATA[5]                                           4.8806
 RE_RDY                                                4.2921
 REG_DATA[6]                                           3.0829
 REG_ADDR[4]                                           2.9131
 DX23[22]                                              2.6127
 DX23[11]                                              2.4852
 REG_ADDR[7]                                           2.1484
 DX23[29]                                             -0.0000

1
pt_shell> report_si_bottleneck -cost_type delta_delay_ratio
 ...
 Bottleneck Cost:  delta_delay_ratio
 net                                                  cost
```

```
 ------------------------------------------------------------
 DX23[17]                                                0.0085
 REG_DATA[5]                                             0.0077
 REG_ADDR[2]                                             0.0076
 RE_RDY                                                  0.0071
 REG_ADDR[7]                                             0.0049
 REG_DATA[6]                                             0.0048
 REG_ADDR[4]                                             0.0045
 DX23[11]                                                0.0040
 DX23[22]                                                0.0039
 DX23[29]                                               -0.0000

1
pt_shell> report_si_bottleneck -cost_type total_victim_delay_bump
 ...
 Bottleneck Cost:  total_victim_delay_bump
 net                                                     cost
 ------------------------------------------------------------
 DX23[17]                                                0.0257
 REG_DATA[5]                                             0.0232
 REG_ADDR[2]                                             0.0224
 RE_RDY                                                  0.0188
 REG_DATA[6]                                             0.0144
 REG_ADDR[4]                                             0.0133
 REG_ADDR[7]                                             0.0129
 DX23[11]                                                0.0124
 DX23[22]                                                0.0118
 DX23[29]                                                0.0000

1
pt_shell> report_si_bottleneck -cost_type delay_bump_per_aggressor
 ...
 Bottleneck Cost:  delay_bump_per_aggressor
 net                                                     cost
 ------------------------------------------------------------
 n24331                                                  0.0257
 _CG_RDY_IN                                              0.0232
 n26710                                                  0.0224
 REG_ADDR[2]                                             0.0188
 REG_ADDR[8]                                             0.0144
 n21337                                                  0.0133
 n18431                                                  0.0129
 BA_190                                                  0.0124
 n17799                                                  0.0118
 AS0_150                                                 0.0000

1
```

Feedback

See Also

- get_si_bottleneck_nets
- insert_buffer
- report_bottleneck
- set_coupling_separation
- size_cell
- report_default_significant_digits

---

report_si_delay_analysis

Generates a report of user coupling information on nets for crosstalk delay analysis.

Syntax

int *report_si_delay_analysis*

```
[-ignored_arrival]
[-excluded]
[-coupling_separated]
[-disabled_statistical]
[-nosplit]
[nets]
```

Data Types

nets	list

Arguments

`-ignored_arrival`

Reports the list of nets you have specified to be analyzed with infinite window.

`-excluded`

Reports the list of nets excluded by you from crosstalk analysis as victim nets or aggressor nets. Also, specifies the analysis type for which the victim/aggressor nets are excluded. It indicates whether the information is set globally on a victim for all aggressors, or globally on a aggressor for all its victims, or between a particular victim and aggressor net. Indicates the type of analysis - minimum path victim falling, minimum path victim rising, maximum path victim falling and maximum path victim rising analysis. The combination of analysis types for which the net is excluded is reported.

`-coupling_separated`

Reports the list of nets on which you have specified coupling separation constraint. It reports the aggressor nets which are separated from all their victims, the victim nets that are separated from all their aggressors, and separation applied between a particular pair of victim and aggressor net.

`-disabled_statistical`

Reports the nets which you have disabled for statistical analysis when the nets are considered as composite aggressor.

`-nosplit`

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`nets`

Limits a list of nets in the current design for which the report needs to be generated.

Description

Produces a report showing coupling information specified by you on nets for crosstalk analysis.

The *report_si_delay_analysis* command allows you to view all of the sets of nets on which you have indicated the default crosstalk analysis behavior to be ignored and the behavior specified by you to take precedence. For every pair of nets, the type of behavior and analysis for which the default behavior is ignored, is reported. This command also specifies whether the information is specified for the net as a victim or aggressor.

Any combination of the options can be used to generate the corresponding report information. If no options are specified, the command reports all the information on all the nets.

This command reports the behavior set by the *set_si_delay_analysis*, *remove_si_delay_analysis*, *set_coupling_separation*, *remove_coupling_separation*, *set_si_delay_disable_statistical* and *remove_si_delay_disable_statistical* commands.

Examples

The following example shows how the excluded option can be used to report all the pairs of victims or aggressors that have been excluded from crosstalk analysis.

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets CLK_NET*]
1
pt_shell> set_si_delay_analysis -exclude -agressors [get_nets AGG_NET*]
1
```

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets V_NET]
-aggressors [get_nets A_NET]
1

pt_shell> report_si_delay_analysis -excluded

****************************************
Report : report_si_delay_analysis
        -excluded
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignored arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net           Delay Analysis
 attributes
-------------------------------------------------------------------------
----
CLK_NET1                *                       E{ mr mf Mr Mf }
CLK_NET2                *                       E{ mr mf Mr Mf }
CLK_NET3                *                       E{ mr mf Mr Mf }
*                       AGG_NET1                E{ mr mf Mr Mf }
*                       AGG_NET2                E{ mr mf Mr Mf }
V_NET                   A_NET                   E{ mr mf Mr Mf }
1
```

The '*' indicates all victim nets/ aggressor nets respectively.

The following example shows how to report all nets in a design which you have specified to be analyzed as infinite window.

```
pt_shell> set_si_delay_analysis -ignore_arrival [get_nets LOGIC0]
1
pt_shell> report_si_delay_analysis -ignored_arrival

****************************************
Report : report_si_delay_analysis
        -ignored_arrival
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************
```

```
Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Delay Analysis
 attributes
-------------------------------------------------------------------------
-----
LOGIC0                  *                        I
*                       LOGIC0                   I
1
```

The following example shows how to report all nets in a design on which you have specified a coupling separation constraint.

```
pt_shell> set_coupling_separation [get_nets LOGIC1]
1
pt_shell> report_si_delay_analysis -coupling_separated

****************************************
Report : report_si_delay_analysis
        -coupling_separated
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Delay Analysis
 attributes
-------------------------------------------------------------------------
-----
LOGIC1                  *                        P
*                       LOGIC1                   P
1
```

Feedback

The following example shows how to report all nets in a design which you have specified to be disabled from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_delay_disable_statistical [get_nets LOGIC1]
1
pt_shell> report_si_delay_analysis -disabled_statistical

****************************************
Report : report_si_delay_analysis
        -disabled_statistical
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignored arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net           Delay Analysis
 attributes
-------------------------------------------------------------------------
-----
LOGIC1                  *                       T
*                       LOGIC1                  T
1
```

The following example shows how to limit the report to a list of nets in a design on which you have specified any coupling information for crosstalk delay analysis.

```
pt_shell> report_si_delay_analysis [get_nets V1]

****************************************
Report : report_si_delay_analysis
        -coupling_separated
        -disabled_statistical
        -excluded
        -ignored_arrival
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     mr - minimum rise analysis
```

```
     mf - minimum fall analysis
     Mr - Maximum rise analysis
     Mf - Maximum fall analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net           Delay Analysis
 attributes
-------------------------------------------------------------------------
-----
V1                      *                       I
*                       V1                      I T
V1                      V2                      E{ mr mf Mr Mf }
1
```

See Also

- set_si_delay_analysis
- remove_si_delay_analysis
- set_coupling_separation
- remove_coupling_separation
- set_si_delay_disable_statistical
- remove_si_delay_disable_statistical

---

report_si_double_switching

Reports the double switching violation detected in the design.

Syntax

status *report_si_double_switching*

```
[-clock_network]
[-rise]
[-fall]
[-nosplit]
[nets]
```

Data Types

```
nets     list
```

Feedback

Arguments

`-clock_network`

Reports the double switch violations for the clock network only, even if the *si_xtalk_double_switching_mode* variable is set to *clock_network* or *full_design*.

`-rise`

Reports the double switch violations for the rising victim.

`-fall`

Reports the double switch violations for the falling victim. If you use neither the *-rise* nor *-fall* option, the net is reported only one time with the smallest slack.

`-nosplit`

Suppresses line splitting if the line exceeds 80 characters.

nets

Reports for the specific victim nets. By default, all nets are reported.

Description

This command reports the victim nets with double switching violations. It reports the victim net name, actual bump height, the required bump height, and the double switching slack.

To use this command, you must enable double switching detection by setting the *si_xtalk_double_switching_mode* variable before the *update_timing* command.

Examples

The following example reports all double switching violations in the design for both rise and fall.

```
pt_shell> report_si_double_switching -nosplit -rise -fall

****************************************
Report : si_double_switching
        -nosplit
        -rise
        -fall
 ...
****************************************

  Victim    Switching   Actual        Required      Double Switching
  Net       Direction   Bump Height   Bump Height   Slack
  ------    ---------   -----------   -----------   ----------------
  I2        max_rise    0.69          0.42          -0.26 (VIOLATING)
  I2        max_fall    0.51          0.49          -0.02 (VIOLATING)
```

Feedback

See Also

- si_enable_analysis
- si_xtalk_double_switching_mode

---

report_si_noise_analysis

Generates a report of user coupling information on nets for crosstalk noise analysis.

Syntax

int *report_si_noise_analysis*

```
[-ignored_arrival]
[-excluded]
[-coupling_separated]
[-disabled_statistical]
[-nosplit]
[nets]
```

Data Types

nets	list

Arguments

`-ignored_arrival`

Reports the list of nets you have specified to be analyzed with infinite window for noise analysis.

`-excluded`

Reports the list of nets excluded by you from noise analysis as victim nets or aggressor nets. Also specifies the analysis type for which the victim/aggressor nets are excluded. The analysis type may be above low rail, below low rail, above high rail or below high rail. The combination of analysis types for which the net is excluded is reported. It indicates whether the information is set globally on a victim for all aggressors, or globally on a aggressor for all its victims, or between a particular victim and aggressor net.

`-coupling_separated`

Reports the list of nets on which you have specified coupling separation constraint. It reports the aggressor nets which are separated from all their victims, the victim nets which have been separated from all their aggressors, and coupling separation applied between a particular victim and aggressor net.

`-disabled_statistical`

Reports the nets which you have disabled for statistical analysis when the nets are considered as composite aggressor.

`-nosplit`

Prevents line splitting and facilitates writing software to extract information from the report output. If you do not use this option, most of the design information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`nets`

Limits a list of nets in the current design for which the report needs to be generated.

Description

Produces a report showing coupling information specified by you on nets for noise analysis.

The *report_si_noise_analysis* command allows you to view all the set of nets on which you have indicated the default noise analysis behavior to be ignored and the behavior specified by you to take precedence. For every pair of nets, the type of behavior and analysis for which the default behavior is ignored, is reported. This command also specifies whether the information is specified for the net as a victim or aggressor.

Any combination of the options can be used to generate the corresponding report information. If no options are specified, the command reports all the information on all the nets.

This command reports the behavior set by the *set_si_noise_analysis*, *remove_si_noise_analysis*, *set_coupling_separation*, *remove_coupling_separation*, *set_si_noise_disable_statistical* and *remove_si_noise_disable_statistical* commands.

Examples

The following example shows how the excluded option can be used to report all the pairs of victims or aggressors that have been excluded from noise analysis.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets CLK_NET*]
1
pt_shell> set_si_noise_analysis -exclude -agressors [get_nets AGG_NET*]
1
pt_shell> set_si_noise_analysis -exclude -victims [get_nets V_NET]
-aggressors [get_nets A_NET]
1

pt_shell> report_si_noise_analysis -excluded

****************************************
```

```
Report : report_si_noise_analysis
         -excluded
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  R - Net is Reselected for each noise iteration
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Noise Analysis
 attributes
-------------------------------------------------------------------------
----
CLK_NET1                *                        E{ al ah bl bh }
CLK_NET2                *                        E{ al ah bl bh }
CLK_NET3                *                        E{ al ah bl bh }
*                       AGG_NET1                 E{ al ah bl bh }
*                       AGG_NET2                 E{ al ah bl bh }
V_NET                   A_NET                    E{ al ah bl bh }
1
```

The '*' indicates all victim nets/ aggressor nets respectively.

The following example shows how to report all nets in a design which you have specified to be analyzed as infinite window.

```
pt_shell> set_si_noise_analysis -ignore_arrival [get_nets LOGIC0]
1
pt_shell> report_si_noise_analysis -ignored_arrival

****************************************
Report : report_si_noise_analysis
         -ignored_arrival
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
```

```
  I - Net has Ignore arrival
  P - Net has couPling constraint
  R - Net is Reselected for each noise iteration
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net           Noise Analysis
 attributes
-------------------------------------------------------------------------
----
LOGICO                  *                       I
*                       LOGIC0                  I
1
```

The following example shows how to report all nets in a design on which you have specified a coupling separation constraint.

```
pt_shell> set_coupling_separation [get_nets LOGIC1]
1
pt_shell> report_si_noise_analysis -coupling_separated

****************************************
Report : report_si_noise_analysis
         -coupling_separated
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
  R - Net is Reselected for each noise iteration
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net           Noise Analysis
 attributes
-------------------------------------------------------------------------
-----
LOGIC1                  *                       P
*                       LOGIC1                  P
1
```

The following example shows how to report all nets in a design which you have specified to be disabled from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_noise_disable_statistical [get_nets LOGIC1]
1
```

```
pt_shell> report_si_noise_analysis -disabled_statistical

****************************************
Report : report_si_noise_analysis
         -disabled_statistical
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
  I - Net has Ignored arrival
  P - Net has couPling constraint
  T - Net is disabled for sTatistical analysis

Victim net              Aggressor net            Noise Analysis
 attributes
-------------------------------------------------------------------------
----
LOGIC1                  *                        T
*                       LOGIC1                   T
1
```

The following example shows how to limit the report to a list of nets in a design on which you have specified any coupling information for noise noise analysis.

```
pt_shell> report_si_noise_analysis [get_nets V1]

****************************************
Report : report_si_noise_analysis
         -coupling_separated
         -disabled_statistical
         -excluded
         -ignored_arrival
         -reselected
Design : TestBH
Version: X-2005.06
Date   : Fri Feb 11 11:28:09 2005
****************************************

Attributes:
  E - Net is Excluded by user for
     al - above low analysis
     ah - above high analysis
     bl - below low analysis
     bh - below high analysis
  I - Net has Ignore arrival
  P - Net has couPling constraint
```

```
  T - Net is disabled for sTatistical analysis

Victim net                Aggressor net             Noise Analysis
 attributes
-------------------------------------------------------------------------
----
V1                        *                         I
*                         V1                        I T
V1                        V2                        E{ al ah bl bh }
1
```

See Also

- set_si_noise_analysis
- remove_si_noise_analysis
- set_coupling_separation
- remove_coupling_separation
- set_si_noise_disable_statistical
- remove_si_noise_disable_statistical

---

report_signal_em_violation

Reports the electromigration rule violations based on the signal EM analysis result.

Syntax

string *report_signal_em_violation*

```
[nets]
[-clear]
[-verbose]
[-report_ratio]
[-ratio_threshold threshold]
[-violation_type type]
[-sort_violation_type type]
[-sort_by value]
```

Data Types

```
nets      list
threshold float
type      string
value     string
```

Arguments

nets

Specifies a list of signal nets to be reported for electromigration violations. Items within the list can be net name patterns or collections. The default is to report all violation nets in the current *check_signal_em* result.

-verbose

print detail information.

-clear

clear *check_signal_em* result from current PrimePower session to release memory.

-report_ratio

report EM ratio (EM ratio = current / current limit) instead of current value.

-ratio_threshold *threshold*

report violation nets and resistors which EM ratio are larger than threshold. (EM ratio = current / current limit). The report is generated from current *check_signal_em* result. Default value is 1. If *ratio_threshold* is 0, all signal nets and resistors analyzed by *check_signal_em* will be reported. If the value is between 0 and 1, this command reports nets from current *check_signal_em* result. Some nets which don't have EM violations may not be reported, if they are not in *check_signal_em* result or filtered. Please see *check_signal_em -skip_filter* for more net filtering information.

-violation_type *type*

report specified violation type. The valid types are all (default), average, rms, and peak.

-sort_violation_type *type*

report resistors sorted by specified violation type. The valid types are average, rms (default), and peak. The default value sorts resistors by the rms type. This option is valid when *violation_type* is all. If *violation_type* is peak, then *sort_violation_type* will be peak automatically.

-sort_by *value*

report resistors sorted by specified value. The valid values are current (default) and ratio.

Feedback

Description

The command reports the electromigration rule violations for the nets included in the current *check_signal_em* result. You must run the command after the *check_signal_em* command.

Electromigration analysis is used to determine whether the current density at a location in the circuit exceeds the defined electromigration rule limits. The current density limits are intended to control the risk that current-carrying geometries is not degraded by excessive current flow. High current densities damage the wires and damaged current-carrying wires can ultimately lead to shorts and opens within the metal, causing circuit failure. The silicon device foundries set the current density limit rules of their related metallurgy to ensure reliability. See the *read_signal_em_rules* man page for more details.

By default, this command reports violation nets and violation resistors to have a minimum report size.

Examples

The following example reports electromigration violations on the signal net after *check_signal_em* is complete.

```
pr_shell> report_signal_em_violation
```

The following example reports rms electromigration violations.

```
pr_shell> report_signal_em_violation -violation_type rms
```

The following example reports rms electromigration violations and all resistors (includes no-violation resistors)

```
pr_shell> report_signal_em_violation -violation_type rms -verbose
```

The following example reports rms electromigration violations and sorts resistors by EM ratio.

```
pr_shell> report_signal_em_violation -violation_type rms -verbose
 -sort_by
ratio
```

See Also

- read_signal_em_rules
- set_signal_em_analysis_options
- check_signal_em

Feedback

report_sim_setup

Report the details of the SimLink setup.

Syntax

status *report_sim_setup*

```
[-library]
[-simulator]
```

Arguments

`-library`

Reports only the current *sim_setup_library* configuration.

`-simulator`

Reports only the current *sim_setup_simulator* configuration.

Description

The *report_sim_setup* command reports the details of the current SimLink configuration.

Configure SimLink by using the *sim_setup_library* and *sim_setup_simulator* commands before running this command.

By default, both the library and simulator configurations are reported. You can use command options to limit the report.

Examples

The following example shows a *report_sim_setup -library* report:

```
pt_shell> report_sim_setup -library
****************************************
Report : report_sim_setup
****************************************

  Library : ncx
    Sub circuit : /WORKING_DIRECTORY/run/subckt_dir_211019183614_453
    File name pattern : ^%s(\..*)?$

 Header : /
u/cltmgr/xtalk/si_lib_gen/unit/lsi_28nm_sch30p/model/header.spi_new_2019
```

The following example shows a *report_sim_setup -simulator* report:

```
pt_shell> report_sim_setup -simulator
****************************************
Report : report_sim_setup
****************************************
```

```
  Simulator : /global/apps/hspice_2018.09-SP2-2/hspice/bin/hspice
  Sim options : .option ingold=2 numdgt=6 measdgt=6 brief statfl=1 acct=0
 autostop runlvl=5
  Work dir : /WORKING_DIRECTORY/run/sim_dir
  Preserve logs : all
  Pre filter : none
```

The following example shows a *report_sim_setup* without any options:

```
pt_shell> report_sim_setup
****************************************
Report : report_sim_setup
****************************************

  Simulator : /global/apps/hspice_2018.09-SP2-2/hspice/bin/hspice
  Sim options : .option ingold=2 numdgt=6 measdgt=6 brief statfl=1 acct=0
 autostop runlvl=5
  Work dir : /WORKING_DIRECTORY/run/sim_dir
  Preserve logs : all
  Pre filter : none

  Library : ncx
    Sub circuit : /WORKING_DIRECTORY/run/subckt_dir_211019183614_453
    File name pattern : ^%s(\..*)?$

 Header : /
u/cltmgr/xtalk/si_lib_gen/unit/lsi_28nm_sch30p/model/header.spi_new_2019
```

See Also

- sim_setup_library
- sim_setup_simulator

---

report_sms_scenarios

Reports the active status of SMS scenarios for selective SMVA or SMC analysis.

Syntax

string *report_sms_scenarios*

```
[-disabled sms_scenarios_list]
```

Data Types

```
sms_scenarios_list      collection
```

Feedback

Arguments

`-disabled` *sms_scenarios_list*

Specifies the report type. The command reports the disabled SMS scenarios. This option is only available with SMVA or SMC analysis.

Description

The *report_sms_scenarios* command reports the active status of SMS scenarios for selective SMVA or SMC analysis. It reports a summarized status of the SMS scenarios configured using the *configure_sms_scenarios* command. This command is only available with SMVA or SMC analysis.

Examples

The following example specifies 3 voltage reference names {high,low,typ} for each one of the supply groups SN1 and SN2. PrimeTime automatically considers all the relevant combinations of voltage levels, in this case up to nine prior the use of the configure_sms_scenarios command. The example disables all SMS scenarios for which the supply group SN1 is at 'typ' voltage and SN2 is at 'typ' voltage.

```
pt_shell> set_voltage_levels SN1 -reference_names {high low typ}
pt_shell> set_voltage -o SN1 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN1 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN1 -reference_name typ 0.9 -min 1.0
pt_shell> set_voltage_levels SN2 -reference_names {high low typ}
pt_shell> set_voltage -o SN2 -reference_name low 0.8 -min 0.9
pt_shell> set_voltage -o SN2 -reference_name high 1 -min 1.1
pt_shell> set_voltage -o SN2 -reference_name typ 0.9 -min 1.0

pt_shell> report_sms_scenarios -disabled
****************************************
Report : report_sms_scenarios
 -disabled
Design : top
Version: T-2022.03
Date   : Mon Oct 28 17:13:07 2022
****************************************

All scenarios are currently non-ignored.

pt_shell> configure_sms_scenarios -disable [get_sms_scenario
 -supply_group SN1 -reference_name typ]
pt_shell> configure_sms_scenarios -disable [get_sms_scenario
 -supply_group SN2 -reference_name typ]

pt_shell> report_sms_scenarios -disabled
****************************************
Report : report_sms_scenarios
 -disabled
```

```
Design : top
Version: T-2022.03
Date   : Mon Oct 28 17:13:07 2022
****************************************

  Current Ignored Scenarios:
------------------------------------------------------------
 SN2:high low, SN1:typ
 SN2:typ
```

The following example continues from the previous example and re-activates the SMS scenario having SN1 at 'typ' voltage and SN2 at 'low' voltage.

```
pt_shell> configure_sms_scenarios -enable [get_sms_scenarios \\
            -supply_group SN1 -reference_names typ \\
            -supply_group SN2 -reference_names low]

pt_shell> report_sms_scenarios -disabled
****************************************
Report : report_sms_scenarios
 -disabled
Design : top
Version: T-2022.03
Date   : Mon Oct 28 17:28:06 2022
****************************************

  Current Ignored Scenarios:
------------------------------------------------------------
 SN2:high, SN1:typ
 SN2:typ
```

See Also

- configure_sms_scenarios
- get_sms_scenarios
- set_voltage_levels
- set_voltage

---

report_stc_power_savings

Generates power saving reports by identifying the stability conditions (STC) of the registers.

Syntax

string *report_stc_power_savings*

Feedback

```
[-clock_gate_type      clock_gate_value_type]
[-min_pwr_savings      power_savings]
[-min_reg_width        register_width]
[-csv csv_file]
[-report_nl_names]
[object_list]
```

Data Types

clock_gate_value_type	string
power_savings	float
register_width	integer
csv_file	string
object_list	list

Arguments

`-clock_gate_type` *clock_gate_value_type*

Use this option to specify the type of clock which drives the registers. The clock could be gated type, domain type (non-gated) or both. Default value of *-clock_gate_type* is "all" and other supported values are "gated" and "not_gated".

`-min_pwr_savings` *power_savings*

Use this option to report only those registers whose total power saving, after using the STC power saving technique, is more than or equal to the "power_savings" value. Default value of *-min_pwr_savings* is 0.0.

`-min_reg_width` *register_width*

Use this option to filter the bundled registers so that the bundled registers having width greater than or equal to "register_width" will be considered for power saving estimate. Default value of *-min_reg_width* is 8.

`-csv` *csv_file*

Use this option to specify the path for the csv file for writing the power savings report in a comma seperated value format. A csv file can be viewed in a spreadsheet reader. With this option the report will not be printed on the shell.

`-report_nl_names`

Use this option to report the netlist(gate-level) name of the register. By default, the RTL name or merged RTL name, in case of bus register, is reported for the registers.

object_list

Use this option to apply the STC power savings techniques on the list of registers. Options *-min_reg_width* and *-min_pwr_savings* cannot be supported with this option. Currently only the gate-level register name can be specified.

Feedback

Description

Use the *report_stc_power_savings* command to generate the power savings report after applying the STC(Stability Conditions) power savings techniques on the register/bus registers in the design.

The three important fields that the command prints are the name of registers, potential power savings that could happen using the above technique and the name of Parent Registers whose enable values cause this Stability Condition.

The other fields are: width of register bus, frequency of clock, the register is gated or not, the current and final register gate efficiency percentage, current and final enable probability, current power and the final power.

To use the command, you should first apply the update_power and update_metrics command or apply the compute_metrics command.

Stability Conditions(STC) Power savings Technique

The Stability Condition (STC) technique is one of the sequential clock-gating techniques to find a new enable or strengthen an enable for gated registers. When the upstream register is stable (disabled), no new data can comes to the downstream registers. Hence, the downstream registers can be gated to save the clock power. Let us assume that for a given non-gated input register, if its parents register enable are stable(disabled), then the input register can be gated using the enable of parents, avoid the switching activity of its clock and hence saves the power. This could be done at the power overhead of Integrated clock gate cell and the Delay cell. This command exactly does this and currently it only looks the stability condition of the immediate parents of the registers/bus registers and reports them if there is any power savings.

Register Name in report

The command prints the RTL register name or the merged RTL names in case of bus registers by default. If mapping information does not exist for some registers the gate level name would get printed instead.

To print the gate-level register names instead of the RTL names use the option -report_nl_names.

Examples

The following example shows a list of bundled registers having a width >=8, where the power could be saved by employing STC(Stability Conditions) Techniques. The registers may have either a gated clock or not.

```
pt_shell> report_stc_power_savings -clock_gate_type all
****************************************
Report : report_stc_power_savings
Design : top
Version: S-2021.06-SP5-DEV-20220213
```

```
Date   : Mon Feb 14 13:54:18 2022
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------
Register                              Register            Gated
   Clock Pin           Current             Current             Current
           Final               Final               Final
 Predicted           STC
Name                                  Width               (Y/N)
   Frequency           Register Gating     Enable              Power
           Register Gating     Enable              Power
 Clock Power         Delay

   (MHz)               Efficiency (%)      Probability         (W)
           Efficiency (%)      Probability         (W)
 Savings (W)         Expression
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------
topsc/abc_first_vect[0:63]            64                  Not Gated
   0.5                 0.0                 1.000               7.265e-05
           99.9                0.505               2.829e-06
 6.982e-05           D(FF_E(topup/topupd/abc_r2[0:63]))

topsc/abc_free_vect[0:63]             64                  Not Gated
   0.5                 0.0                 1.000               7.112e-05
           99.9                0.988               2.406e-06
 6.871e-05           D(FF_E(topup/elc/abc_a1[0:63]))

topaf/abc/req_data_s6_ff[33, 37],
topaf/abc/xyz_data[32, 34:36, 38:39]
                                      8                   Gated
   0.5                 99.9                1.000               0.0001693
           100.0               0.0734              0.0001234
 4.588e-05           D(FF_E(topaf/abc/jkl_s3_reg[7:0]))

topaf/xyz_data[48:49, 51:55],
topaf/abc/req_data_s6_ff[50]
                                      8                   Gated
   0.5                 99.9                1.000               0.0001693
           100.0               0.0734              0.0001234
 4.588e-05           D(FF_E(topaf/abc/jkl_s2_reg[7:0]))

topaf/abc/req_data_s6_ff[56],
topaf/xyz_data[57:63]
                                      8                   Gated
   0.5                 99.9                1.000               0.0001693
           100.0               0.0734              0.0001234
 4.588e-05           D(FF_E(topaf/abc/jkl_s2_reg[15:8]))
```

The following example shows a list of bundled registers having a width >=32, where the power could be saved by employing STC(Stability Conditions) Techniques. The registers will have only gated clock.

```
pt_shell> report_stc_power_savings -clock_gate_type gated -min_reg_width
 32
****************************************
Report : report_stc_power_savings
Design : abc
Version: S-2021.06-SP5-DEV-20220213
Date   : Mon Feb 14 14:13:19 2022
****************************************

--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------
Register                              Register            Gated
   Clock Pin           Current             Current             Current
           Final               Final               Final
 Predicted           STC
Name                                  Width               (Y/N)
   Frequency           Register Gating     Enable              Power
           Register Gating     Enable              Power
 Clock Power         Delay

   (MHz)               Efficiency (%)      Probability         (W)
           Efficiency (%)      Probability         (W)
 Savings (W)         Expression
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------
abc_de_0/xyz/st2_Mj1[0:31]       32                  Gated
  333.3               69.4                0.306               1.853e-05
           77.9                0.221               1.339e-05
 5.139e-06           D(FF_E(abc_de_0/xyz/mn11[0:15]


                      abc_de_0/xyz/mn21[0:15]))

abc_de_0/xyz/st2_Mi1[0:31]       32                  Gated
  333.3               69.4                0.306               1.789e-05
           77.9                0.221               1.314e-05
 4.755e-06           D(FF_E(abc_de_0/xyz/mn11[0:15]


                      abc_de_0/xyz/mn21[0:15]))
```

```
abc_de_0/xyz/st2_A0[0:31]         32                    Gated
  333.3                88.9                  0.111                 7.572e-06
           94.7                  0.0537               4.653e-06
 2.919e-06           D(FF_E(abc_de_0/xyz/mn10[0:15]


                       abc_de_0/xyz/mn20[0:15]


                       abc_de_0/xyz/cmd_r0[11:18]))

abc_de_0/xyz/st4_Q[0:39]          40                    Gated
  333.3                87.0                  0.131                 9.858e-06
           89.3                  0.107                7.575e-06
 2.282e-06           D(FF_E(abc_de_0/xyz/mn3_acc0j[0:39]


                       abc_de_0/xyz/mn3_acc0i[0:39]),


                  FF_E(abc_de_0/xyz/mn3_acc1j[0:39]


                       abc_de_0/xyz/mn3_acc1i[0:39]),


                  FF_E(abc_de_0/xyz/cmd_r2[4:11]))

abc_de_0/ram_ag_1/xbar_cmd_r[0:39] 40                   Gated
  333.3                94.8                  0.053                 2.646e-06
           99.1                  0.00852              1.523e-06
 1.123e-06           D(FF_E(abc_de_0/idx0[4],


                       abc_de_0/ad0/buf2[0:2, 5:39],


                       abc_de_0/ad0/data_out0[3]))
```

See Also

- compute_metrics
- update_power

Feedback

- update_metrics
- report_power
- report_rtl_metrics
- report_est_power_savings
- report_user_def_clock_savings

---

report_supply_group

Reports existing supply groups.

Syntax

```
status report_supply_group
  [supply_group_list]
```

Data Types

```
supply_group_list           string
```

Arguments

supply_group_list

Specifies supply groups to report.

Description

The *report_supply_group* command reports detailed information about the supply groups, including the name, voltage levels defined by the *set_voltage_levels* command, and voltage values that are set per voltage level by the *set_voltage* command.

Examples

The following example reports the information of all supply groups:

```
prompt> report_supply_group
------------------------------------------------------------------------
------
    Supply group :      SN1
    Voltage level :     k
    Max-delay Voltage : Not assigned yet
    Min-delay Voltage : Not assigned yet
    Voltage level :     l
    Max-delay Voltage : 0.8000
    Min-delay Voltage : 0.9000
    Voltage level :     m
    Max-delay Voltage : Not assigned yet
    Min-delay Voltage : Not assigned yet
```

Feedback

```
------------------------------------------------------------------------
------
    Supply group :      SN2
    Voltage level :     Not assigned yet

1
```

See Also

- get_supply_groups
- set_voltage_levels

---

report_supply_net

Reports existing supply nets.

Syntax

```
status report_supply_net
   [-domain domain_name]
   [-scope scope_name]
```

Data Types

```
domain_name           string
scope_name            string
```

Arguments

`-domain domain_name`

Specifies domain-dependent supply nets that exist in a particular power domain. The domain must exist.

`-scope scope_name`

Specifies for which scope you want to define the supply nets that are reported. If this is not specified, the command reports all the power nets in the current scope.

Description

The *report_supply_net* command provides detailed information about all supply nets associated with a specified power domain or scope. If you specify a domain, the report shows only the domain-dependent supply nets that exist in the domain. The information of supply nets include:

- The full name
- All power domains that it is created in

Feedback

- All voltage information that is explicitly set on this net by the *set_voltage* command
- All domains for which the net is the primary power or ground net

Examples

The following example reports the information of all supply nets in the current scope.

```
prompt> report_supply_net
  Total of 1 supply nets defined.
------------------------------------------------------------------------
    Supply Net :        i2/net_i2
    Scope :             i2
    Power Domains :
    Supply Ports  :     N/A
    PG_power_pin  :     N/A
    Max-delay Voltage : 1.08
    Min-delay Voltage : 1.3
    Resolve type :      unresolved
1
```

See Also

- connect_supply_net
- create_power_domain
- create_supply_net
- get_power_domains
- set_domain_supply_net
- set_scope

---

report_supply_set

Reports all supply sets defined in the design.

Syntax

status *report_supply_set*

Arguments

None.

Description

The *report_supply_set* command reports detailed information about the supply sets defined in the design. The report for a supply set includes the following information: its full

name, the scope in which it was created, and the pairs of function and associated supply net.

This command returns 1 on success, otherwise a 0 returns.

Examples

The following example reports information about the primary_sset supply set created in the current scope.

```
prompt> create_power_domain top_domain
top_domain

prompt> create_supply_set sset \\
          -function {power VDD} \\
          -function {ground VSS}
sset

prompt> report_supply_set

  Total of 2 supply sets defined for design 'top'.

--------------------------------------------------------------------------------
------
    Supply Set : top_domain.primary
    Scope      : <top level>
    Function   : power, supply net association : top_domain.primary.power
    Function   : ground, supply net association :
 top_domain.primary.ground
    Associated : -
--------------------------------------------------------------------------------
------
    Supply Set : sset
    Scope      : <top level>
    Function   : power, supply net association : VDD
    Function   : ground, supply net association : VSS
1
```

See Also

- create_supply_set

---

report_switching_activity

Reports the statistics on the switching activity annotation and signal probability on the current design or instance.

Syntax

integer *report_switching_activity*

```
[-cells cells_list]
[-list_not_annotated]
[-list_annotated]
[-list_low_activity]
[-list_by_source source]
[-base_clock clk]
[-average_activity]
[-coverage]
[-hierarchy]
[-show_pin]
[-sort_by hier | net_toggle_rate | name]
[-toggle_limit limit]
[-exclude exclusion_group]
[-include_only_user_group  inclusion_user_group]
[-include_only inclusion_group]
[-only_related_clock clock]
[-include_mapping_types]
```

Data Types

cells_list	list
source	string
clk	string
limit	integer
exclusion_group	string
inclusion_user_group	string
inclusion_group	string
clock	clock

Arguments

`-cells` *cells_list*

Indicates that switching activity annotation is to be reported only for the specified *cells_list*.

`-list_not_annotated`

Reports a list of design nets that have no user-defined switching activity annotation. When specified, the report does not include constant value nets (logic one/zero nets). Note that such nets are annotated with the default switching activity if they are not user-annotated.

`-list_annotated`

Reports a list of design nets that do have user-defined switching activity annotation.

`-list_low_activity`

Reports a list of nets in the design with fewer toggles than the value defined using the -toggle_limit option. Use the list report to debug which nets are not being exercised by the simulation test bench.

`-list_by_source` *source*

Reports a list of nets that have switching activity information from a given source.

Possible sources are any of the following:

[annotated, file, propagated, default, set_switching_activity, set_case_analysis, create_clock, implied, no_switching_activity]

Here *annotated* means any annotated (not propagated) source. The *file* source means any source with from an activity file (either VCD or SAIF file).

`-base_clock` *clk*

When the -average_activity option is specified, the average activity over nets is reported. The averaged toggle rates reported are by default with respect to 1ns. When the -base_clock option is used, the reported averaged toggle rates are with respect to the period of the clock *clk*. If the reported toggle rate is 0.5, and the period of the base clock is 3ns, then on average during simulation, there are 0.5 toggles every 3ns, or 1 toggle every 6ns.

`-average_activity`

Produces a report that averages toggle rates over the nets in the design. When combined with the -hierarchy option, the average switching activity is computed for each subblock in the design.

It is possible using the filter options, -exclude or -include_only, to get a report of average toggle rates over a subset of the nets in the design. This can be useful for instances to get the average toggle rate for annotated nets, or even the average toggle rate for annotated nets driven by the sequential cells.

`-coverage`

Creates e a summary report about the nets and the design that have switching activity information, but have few toggles. Coverage is defined as the percentage of nets with more than the value defined by the -toggle_limit *limit* option. You can use this report to verify that the switching activity from simulation or propagation is properly exercising the design. When used with the -hierarchy option, you can use the report to make sure that each block in the design is properly exercised.

`-hierarchy`

Includes the information about subblocks in the design. The flag cannot be used with the list generating options for the command.

`-show_pin`

Specifies that driver pins as well as nets are shown in certain net lists. The lists generated by the -list_not_annotated, -list_annotated, -list_low_activity, and -list_by_source options are affected. Pins are not shown for multidriver nets.

`-sort_by` *`hier | net_toggle_rate | name`*

When used with any of the -hierarchy option with the default report or with the -average_activity or -coverage option, the hierarchical blocks in the report are sorted either by hierarchy (depth first), averaged toggle rate, or name.

When used with any of the list options (that is, the -list_low_activity, -list_by_source, -list_not_annotated, or -list_annotated options), the list is sorted by toggle rates or by names. For the list reports, the -sort_by *hier* option is not accepted.

`-toggle_limit` *`limit`*

Sets the lower limit for what is considered to be few toggles. This limit is used by the -coverage and the -list_low_activity options. The default limit is 1 toggle.

The value of the argument to the -toggle_limit option represents the maximum number of toggles considered to be low activity. The number of toggles on a net is defined to be the toggle rate times the simulation duration. For vector free power analysis (where no SAIF or VCD file has been used), the default simulation duration is 10000 ns. If a SAIF file has been used, the duration is taken from the SAIF file. If a VCD file has been used to annotate activity, the duration is the duration of the VCD simulation.

`-exclude` *`exclusion_group`*

Filters out the nets from consideration before generating any of the reports. Filtering occurs before toggle rate averaging, coverage percentage computations, and list generation. Possible exclusion groups are any of the following:

[*sequential, combinational, primary_inputs, black_boxes, three_state, memory, clock_gate, clock, low_activity, high_activity, rtl*]

A net is defined as sequential if it is driven by a sequential cell, same with combinational, primary_inputs, etc.

The RTL exclusion group refers to the nets driven either by sequential cell outputs, black box outputs, primary inputs, or memories. The RTL group is mainly provided as a shorthand for writing out all of these.

The clock_gate exclusion group refers to ICGs only, not other types of clock gates. Other possible exclusion groups are defined by the source of the activity information:

[*annotated, file, propagated, default, set_switching_activity, set_case_analysis, create_clock, implied, no_switching_activity*]

Here *annotated* means any annotated (not propagated) source. The *file* source means any source from an activity file (usually VCD or SAIF files).

To exclude multiple groups, the *exclusion_group* argument can be a list with multiple groups in the list.

Note: Currently, the "sequential" group should not include the nets driven by clock gates. In the C-2006.12 release, the group included such nets.

`-include_only_user_group` *`inclusion_user_group`*

Includes only the nets under the *inclusion_user_group* or the groups defined by the *report_switching_activity* command.

pt_shell> *report_switching_activity -include_only_user_group {user_group}*

`-include_only` *`inclusion_group`*

Includes the nets under the \inclusion_group when generating any of the reports. Filtering occurs before toggle rate averaging, coverage percentage computations, and list generation.

Possible inclusion groups are any of the following: [*sequential, combinational, primary_inputs, black_boxes, three_state, memory, clock_gate, clock, low_activity, high_activity, rtl*].

A net is defined as sequential if it is driven by a sequential cell, same with combinational, primary_inputs, etc. The clock_gate inclusion group refers to ICGs only, not other types of clock gates.

Other possible inclusion groups are defined by the source of the activity information, such as: [*annotated, file, propagated, default, set_switching_activity, set_case_analysis, create_clock, implied, no_switching_activity*]

Here *annotated* means any annotated (not propagated) source. The *file* source means any source from an activity file (usually a VCD or a SAIF file).

To include multiple groups, use the *inclusion_group* argument to specify multiple groups in the list. In this case, only nets that are in one of the listed groups are included in the report.

To include only the nets that are in several groups, use the *&* operator. This operator has low precedence; lists are or'ed first, then and'ed. For instance, to

include only the nets that are driven by RTL points or three_state buffers and have no switching activity, use the following command:

pt_shell> *report_switching_activity -include_only {rtl, three_state & no_switching_activity}*

To invert the selection, the *!* operator can be used. For example, one way to include only the nets that are driven by RTL points and have switching activity would be the following:

pt_shell> *report_switching_activity -include_only {rtl & !no_switching_activity}*

There may be other ways (possibly using the -exclude option) to get this same set of nets included in the report.

`-only_related_clock` *clock*

If specified, nets that have a base_clock other than *clock* are filtered out before the results of the report are included. This option can affect the overview, coverage, or the average toggle report. Also, this option can affect the output lists. Such reports consider only the nets with the given base_clock.

`-include_mapping_types`

When specified, the "Annotated from File" column in the report is split into three columns: Name mapping file, Exact name mapping, and Build-in name mapping, which list the number of netlist objects mapped using each of the techniques respectively.

Description

The command report the switching activities on the nets in the current design or instance.

The default report generated by the *report_switching_activity* command gives an overview of switching activity information in the current instance. The report lists the percentage of nets with switching activity and signal probability that is user-annotated, default-annotated, and propagated (if any).

Switching activity and signal probability can be annotated by using the *set_switching_activity* and *read_saif* commands, or by using event-simulation data using the the *read_vcd* command. The *create_clock* and *set_case_analysis* commands also affect switching activities.

Note that the *read_vcd* command, when used with a named pipe or the *-pipe* option, does not immediately annotate activities on the design. The annotations happen later during the *update_power* command. Because of this, the *report_switching_activity* command does not report activity change in this situation until the *update_power* command has been run.

During power calculations, PrimePower estimates the switching activity of objects that are not user-annotated by propagating the user or default annotated switching activity

values. These objects are reported as propagated after the *update_power* command has been run, but as not propagated prior to running the *update_power* command. In general, propagated switching activity is less accurate than switching activity on objects derived by simulation, so the percentage values in the user-annotated column of the report generated by the *report_switching_activity* command are an indication of the accuracy of power reports.

If the *power_analysis_mode* command has been set to *averaged* or *leakage_variation* and if a VCD file has been read with the *read_vcd* command without the -pipe option, the tool annotates the activity from the VCD file prior to reporting the switching activity. In addition, unannotated objects where the activity can be derived accurately without random vector simulation (for example, clock nets, Qbar pins where the Q pin is annotated, and constant nets) is reported by the *report_switching_activity* command as having *implied\P activity. Activity propagation for other unannotated nodes does not occur until the update_power command is run.*

If the *power_analysis_mode* has been set to *time_based*, the *get_switching_activity* command does not have access to toggle rate information prior to running the *update_power* command. The command reports whether a net is uninitialized or whether it is annotated from the VCD file, but since the VCD file has not been read yet, the toggle rate and other values are reported as -1.

In some instances, you may have incomplete annotation from simulation. In this case, one or more nets may be unannotated after reading the activity file. Part of the purpose of the *report_switching_activity* command is to help you locate these annotation problems. If possible, you should get complete simulation data. If this is not possible, it can be useful to use the -average_activity option to get average toggle rates in the design for the purpose of patching these annotation problems using the *set_switching_activity* command.

Note that a net is defined as *driverless* if it is not driven by any cell. There is no group defined to report these driverless nets, so added with *combinational* or other groups.

Examples

The following examples report switching accuracy for the current design.

In the following example, the *report_switching_activity* command is used after the *read_saif* command.

```
pt_shell> report_switching_activity

****************************************
Report : Switching Activity

Design : mac
Version: B-2008.12-Beta2-DEV
Date   : Thu Oct  2 09:29:40 2008
****************************************
```

Feedback

```
 Switching Activity Overview Statistics for "mac"
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
                  From Activity     From          From          From
                                                   Not
Object Type       File (%)          SSA (%)       SCA (%)       Clock (%)
  Default (%)     Propagated(%)    Implied(%)      Annotated(%)     Total
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
 Nets             1613(99.02%)       0(0.00%)     0(0.00%)      0(0.00%)
  0(0.00%)        0(0.00%)         0(0.00%)         16(0.98%)        1629
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
Nets Driven by
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
Primary Input     67(100.00%)        0(0.00%)     0(0.00%)      0(0.00%)
  0(0.00%)        0(0.00%)         0(0.00%)         0(0.00%)         67
Tri-State         0(0%)              0(0%)        0(0%)         0(0%)
  0(0%)           0(0%)            0(0%)            0(0%)            0
Black Box         0(0%)              0(0%)        0(0%)         0(0%)
  0(0%)           0(0%)            0(0%)            0(0%)            0
Sequential        209(92.89%)        0(0.00%)     0(0.00%)      0(0.00%)
  0(0.00%)        0(0.00%)         0(0.00%)         16(7.11%)        225
Combinational     1337(100.00%)      0(0.00%)     0(0.00%)      0(0.00%)
  0(0.00%)        0(0.00%)         0(0.00%)         0(0.00%)         1337
Memory            0(0%)              0(0%)        0(0%)         0(0%)
  0(0%)           0(0%)            0(0%)            0(0%)            0
Clock Gate        0(0%)              0(0%)        0(0%)         0(0%)
  0(0%)           0(0%)            0(0%)            0(0%)            0
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
 Static Probability Overview Statistics for "mac"
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
                  From Activity     From          From          From
                                                   Not
Object Type       File (%)          SSA (%)       SCA (%)       Clock (%)
  Default (%)     Propagated(%)    Implied(%)      Annotated(%)     Total
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
 Nets             1613(99.02%)       0(0.00%)     0(0.00%)      0(0.00%)
  0(0.00%)        0(0.00%)         0(0.00%)         16(0.98%)        1629
-------------------------------------------------------------------------
-------------------------------------------------------------------------
---
```

```
Nets Driven by
------------------------------------------------------------------------
------------------------------------------------------------------------
---
Primary Input      67(100.00%)        0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         0(0.00%)         67
Tri-State          0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Black Box          0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Sequential         209(92.89%)        0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         16(7.11%)        225
Combinational      1337(100.00%)      0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         0(0.00%)         1337
Memory             0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Clock Gate         0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
------------------------------------------------------------------------
------------------------------------------------------------------------
---
1
```

In the following example, the *report_switching_activity* command is used with the -list_not_annotated option to find the nets that do not have switching activity information. The nets without activity are listed below the overview report.

```
pt_shell> report_switching_activity -list_not_annotated

****************************************
Report : Switching Activity
         -list_not_annotated
Design : mac
Version: B-2008.12-Beta2-DEV
Date   : Thu Oct  2 09:27:39 2008
****************************************

 Switching Activity Overview Statistics for "mac"
------------------------------------------------------------------------
------------------------------------------------------------------------
---
                   From Activity      From          From          From
                                                     Not
Object Type        File (%)           SSA (%)       SCA (%)       Clock (%)
  Default (%)      Propagated(%)    Implied(%)       Annotated(%)     Total
------------------------------------------------------------------------
------------------------------------------------------------------------
---
 Nets              1613(99.02%)       0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         16(0.98%)        1629
```

Feedback

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---
Nets Driven by
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---
Primary Input      67(100.00%)        0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         0(0.00%)         67
Tri-State          0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Black Box          0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Sequential         209(92.89%)        0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         16(7.11%)        225
Combinational      1337(100.00%)      0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         0(0.00%)         1337
Memory             0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Clock Gate         0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---
 Static Probability Overview Statistics for "mac"
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---
                   From Activity      From          From          From
                                                     Not
Object Type        File (%)             SSA (%)       SCA (%)       Clock (%)
  Default (%)      Propagated(%)    Implied(%)       Annotated(%)     Total
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---
 Nets              1613(99.02%)       0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         16(0.98%)        1629
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---
Nets Driven by
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
---
Primary Input      67(100.00%)        0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         0(0.00%)         67
Tri-State          0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Black Box          0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)            0(0%)            0(0%)            0(0%)            0
Sequential         209(92.89%)        0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)         16(7.11%)        225
```

```
Combinational     1337(100.00%)      0(0.00%)      0(0.00%)      0(0.00%)
  0(0.00%)        0(0.00%)         0(0.00%)        0(0.00%)         1337
Memory            0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)           0(0%)            0(0%)           0(0%)            0
Clock Gate        0(0%)              0(0%)         0(0%)         0(0%)
  0(0%)           0(0%)            0(0%)           0(0%)            0
--------------------------------------------------------------------------
--------------------------------------------------------------------------
---


List of nonannotated nets :
        a_r[1]
        a_r[0]
        a_r[14]
        a_r[8]
        a_r[13]
        a_r[4]
        a_r[3]
        a_r[10]
        a_r[12]
        a_r[2]
        a_r[5]
        a_r[11]
        a_r[15]
        a_r[7]
        a_r[9]
        a_r[6]

16 nets with no annotation
1
```

In the following example, the *report_switching_activity* command is used after the *update_power* command. The unannotated nets are now listed as having been propagated.

```
****************************************
Report : Switching Activity

Design : mac
Version: B-2008.12-Beta2-DEV
Date   : Thu Oct  2 09:29:16 2008
****************************************

 Switching Activity Overview Statistics for "mac"
--------------------------------------------------------------------------
--------------------------------------------------------------------------
---
                  From Activity     From          From          From
                                                   Not
Object Type       File (%)            SSA (%)     SCA (%)       Clock (%)
  Default (%)     Propagated(%)    Implied(%)      Annotated(%)     Total
```

```
------------------------------------------------------------------------
------------------------------------------------------------------------
---
 Nets                1613(99.02%)       0(0.00%)      0(0.00%)      0(0.00%)
   0(0.00%)          16(0.98%)        0(0.00%)          0(0.00%)          1629
------------------------------------------------------------------------
------------------------------------------------------------------------
---
Nets Driven by
------------------------------------------------------------------------
------------------------------------------------------------------------
---
Primary Input       67(100.00%)        0(0.00%)      0(0.00%)      0(0.00%)
   0(0.00%)         0(0.00%)         0(0.00%)          0(0.00%)          67
Tri-State           0(0%)              0(0%)         0(0%)         0(0%)
   0(0%)            0(0%)            0(0%)             0(0%)             0
Black Box           0(0%)              0(0%)         0(0%)         0(0%)
   0(0%)            0(0%)            0(0%)             0(0%)             0
Sequential          209(92.89%)        0(0.00%)      0(0.00%)      0(0.00%)
   0(0.00%)         16(7.11%)        0(0.00%)          0(0.00%)          225
Combinational       1337(100.00%)      0(0.00%)      0(0.00%)      0(0.00%)
   0(0.00%)         0(0.00%)         0(0.00%)          0(0.00%)          1337
Memory              0(0%)              0(0%)         0(0%)         0(0%)
   0(0%)            0(0%)            0(0%)             0(0%)             0
Clock Gate          0(0%)              0(0%)         0(0%)         0(0%)
   0(0%)            0(0%)            0(0%)             0(0%)             0
------------------------------------------------------------------------
------------------------------------------------------------------------
---
 Static Probability Overview Statistics for "mac"
------------------------------------------------------------------------
------------------------------------------------------------------------
---
                    From Activity      From          From          From
                                                      Not
Object Type         File (%)             SSA (%)       SCA (%)       Clock (%)
   Default (%)      Propagated(%)    Implied(%)        Annotated(%)     Total
------------------------------------------------------------------------
------------------------------------------------------------------------
---
 Nets               1613(99.02%)       0(0.00%)      0(0.00%)      0(0.00%)
   0(0.00%)         16(0.98%)        0(0.00%)          0(0.00%)          1629
------------------------------------------------------------------------
------------------------------------------------------------------------
---
Nets Driven by
------------------------------------------------------------------------
------------------------------------------------------------------------
---
Primary Input       67(100.00%)        0(0.00%)      0(0.00%)      0(0.00%)
   0(0.00%)         0(0.00%)         0(0.00%)          0(0.00%)          67
Tri-State           0(0%)              0(0%)         0(0%)         0(0%)
   0(0%)            0(0%)            0(0%)             0(0%)             0
```

```
Black Box          0(0%)              0(0%)         0(0%)          0(0%)
  0(0%)            0(0%)            0(0%)           0(0%)            0
Sequential         209(92.89%)        0(0.00%)      0(0.00%)       0(0.00%)
  0(0.00%)         16(7.11%)        0(0.00%)        0(0.00%)         225
Combinational      1337(100.00%)      0(0.00%)      0(0.00%)       0(0.00%)
  0(0.00%)         0(0.00%)         0(0.00%)        0(0.00%)         1337
Memory             0(0%)              0(0%)         0(0%)          0(0%)
  0(0%)            0(0%)            0(0%)           0(0%)            0
Clock Gate         0(0%)              0(0%)         0(0%)          0(0%)
  0(0%)            0(0%)            0(0%)           0(0%)            0
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---
1
```

In the following example, the *report_switching_activity* command is used to get the average toggle rates on the design and subblocks of the design (using the -hierarchy option).

```
pt_shell> report_switching_activity -average -hier

****************************************
Report : Switching Activity
         -average_activity
         -hierarchy
Design : mac
Version: K-2015.12-DEV-151116
Date   : Tue Nov 17 13:13:22 2015
****************************************

 Switching Activities per period is with respect to time period
 9996.000000 ns
 Simulation time for toggle count is 9996 ns
--------------------------------------------------------------------------------
--------------------------
Object                  Toggle Count       Toggle Rate      Glitch Count
 Glitch Rate      # Nets
                        Per Net            Per Period       Per Net
 Per Period       Included
                                           Per Net
 Per Net
--------------------------------------------------------------------------------
--------------------------
mac                     198.74             0.02             0
 0                1633
mult_21                 141.16             0.014            0
 0                937
mult_21/U1/U9720        185.49             0.019            0
 0                261
add_23                  245.41             0.025            0
 0                363
```

```
--------------------------------------------------------------------------
-------------------------
1
```

In the following example, the *report_switching_activity* command is used to get the average toggle rates per clock period on the design and subblocks of the design.

```
pt_shell> report_switching_activity -average -hier -base_clock clk

****************************************
Report : Switching Activity
         -average_activity
         -hierarchy
Design : mac
Version: K-2015.12-DEV-151116
Date   : Tue Nov 17 13:13:47 2015
****************************************

 Switching Activities per period is with respect to Clock 'clk' with
 period 12.000000 ns
 Simulation time for toggle count is 9996 ns
--------------------------------------------------------------------------
-------------------------
Object                 Toggle Count       Toggle Rate      Glitch Count
 Glitch Rate      # Nets
                          Per Net          Per Period       Per Net
 Per Period       Included
                                           Per Net
 Per Net
--------------------------------------------------------------------------
-------------------------
mac                        198.74           0.24             0              0
                 1633
mult_21                    141.16           0.17             0              0
                 937
mult_21/U1/U9720           185.49           0.22             0              0
                 261
add_23                     245.41           0.29             0              0
                 363
--------------------------------------------------------------------------
-------------------------
1
```

In the following example, the *report_switching_activity* command is used with the -coverage option to determine whether or not the simulation testbench from which the SAIF file was derived did a good job in exercising each part of the design. In this example, the simulation did a poor job in exercising the counter_high_bits subblock.

```
pt_shell> report_switching_activity -coverage -hierarchy

****************************************
Report : Switching Activity
```

Feedback

```
        -coverage -hierarchy
Design : counter
Version: V-2004.06-Alpha1
Date   : Mon Mar 29 11:09:34 2004
****************************************
 Switching Activity Coverage for design "counter"
--------------------------------------------------------------------------------
-------------
 Coverage is defined as the percent of nets with at least 1 toggle
--------------------------------------------------------------------------------
-------------
Block                         % Nets Covered    # Nets Covered    Total
 Nets
                                                                   Counted
--------------------------------------------------------------------------------
-------------
counter                        76                76                100
counter_low_bits              100                44                44
counter_high_bits              25                11                44
reset_block                    50                 3                6
overflowoptionect                50                   3                  6
--------------------------------------------------------------------------------
-------------
```

See Also

- read_saif
- reset_switching_activity
- set_switching_activity
- get_switching_activity
- set_rtl_to_gate_name
- update_power
- report_power

---

report_sync_app_vars

Reports application variables synchronized between the manager and worker processes

Syntax

status *report_sync_app_vars*

```
[-skip_header]
[-list_only]
[-nosplit]
[-only_changed_vars]
```

Feedback

Arguments

`-skip_header`

Do not print the report's header.

`-list_only`

Instead of the report, return a list of variables.

`-nosplit`

Prevents line splitting. This can useful for scripts that extract information from the report. By default, the report generates a new line when the text cannot fit in the allotted space in a column.

`-only_changed_vars`

Reports only changed variables.

Description

The *report_sync_app_vars* command reports application variables synchronized between the manager and worker processes.

Distributed Analysis

The command can be executed in the manager process. The command shows the list of variables that will be synchronized from the manager session to worker processes at the start of the next distributed task processing round.

To execute the *report_sync_app_vars* command in all scenarios or partitions, use the *remote_execute* command. For example:

```
remote_execute -verbose {
  report_sync_app_vars
}
```

The report produced by a worker process shows the list of variables and the values that have been set at the start of the current distributed task execution on this worker. Reported values may not match the current values of these variables at the worker if the current task changed values of these variables.

See Also

- report_app_var

---

report_target_library_subset

Reports the target library subset specifications for the top-level design and hierarchical cells.

Feedback

Syntax

status *report_target_library_subset*

```
[-objects objects]
[-top]
[-nosplit]
```

Data Types

objects	list or collection

Arguments

`-objects` *objects*

Reports the subset specifications applied to the specified hierarchical cells.

If you do not specify this option or the *-top* option, the tool reports all specifications explicitly set by the *set_target_library_subset* command.

`-top`

Reports the subset specifications applied to the top-level design.

If you do not specify this option or the *-top* option, the tool reports all specifications explicitly set by the *set_target_library_subset* command.

`-nosplit`

Does not split lines if column overflows.

Description

This command reports the target library subsets applied in the current design. This command reports only subsets that were explicitly set on objects with the *set_target_library_subset* command. This command does not report subsets that are implicitly inherited from a parent hierarchical cell.

Examples

The following example reports the target library subset for a hierarchical cell.

```
prompt> report_target_library_subset -objects [get_cells top/mid]
```

See Also

- set_target_library_subset
- remove_target_library_subset

Feedback

report_threshold_voltage_group

Reports the number and percentage of cells for each threshold voltage group in the design, including any user-specified low-threshold voltage groups.

Syntax

string *report_threshold_voltage_group*

```
[-lvth_groups groups]
[-nosplit]
[-verbose]
[-pattern_priority pattern_list]
[-attribute attribute_name]
[-area]
[-clocks clock_list]
[-rails rails_supply_nets_name]
[-exact_match]
[cell_list]
```

Data Types

groups	list
pattern_list	list
attribute_name	string
cell_list	list
clock_list	list
rails_supply_nets_name	list

Arguments

`-lvth_groups` *groups*

Specifies a list of threshold voltage groups considered low-threshold for reporting purposes. The argument is a list of threshold voltage group identifiers. Each identifier is a string containing only alphanumeric characters and possibly the underscore character. To associate these group identifiers with library cells, set the user-defined attributes named *default_threshold_voltage_group* at the library level or *threshold_voltage_group* at the library cell level.

If you do not use the *-lvth_groups* option, the report does not identify low-threshold-voltage groups.

`-nosplit`

Prevents splitting of long lines that exceed the specified column width.

`-verbose`

Lists all the cells belonging to each threshold voltage group. Without this option, the default summary report shows only the number and percentage of cells in each power group.

`-pattern_priority` *`pattern_list`*

Specifies the list of patterns of cell name (or its attribute value, if using the *-attribute* option) to process for generating the threshold voltage group report.

`-attribute` *`attribute_name`*

Specifies the name of the cell attribute used to match the name patterns specified by the *-pattern_priority* option.

`-area`

Reports the cell area (instead of number of cells) belonging to each threshold voltage group.

`-clocks` *`clock_list`*

Use this option to report only the cells those belong to the clock domains specified by the *clock_list* value.

`-rails` *`rails_supply_nets_name`*

Use this option to specify a list of supply nets under UPF mode or design rails under rail mapping mode for which the report to be shown.

`cell_list`

Specifies a list of cells to report as threshold voltage groups. If this option is omitted, the report covers all of the current design.

`-exact_match`

Use this option to exactly match the cell name pattern specified by the *-pattern_priority* option.

Description

The *report_threshold_voltage_group* command reports the number and percentage of cells in each power group belonging to different threshold voltage groups, for a given list of cells or the whole design. A PrimePower license is required and power analysis must be enabled.

The command identifies the threshold voltage groups to which the cells belong according to the *default_threshold_voltage_group* lib attribute or the *threshold_voltage_group* lib_cell attribute. The lib_cell attribute has priority over the lib attribute in the grouping process.

Clock Domain Reporting

When you specify the *clock_list* value with the *-clocks* option, a report is generated for the cells in the specified clock domains.

The cells in the clock tree network of CLK clock are taken as belonging to CLK clock domain. The cells in a timing path launched by the CLK clock are regarded as belonging

to CLK clock domain. The cells in a timing path with no launch clock, but captured by the CLK clock are regarded as belonging to the CLK clock domain as well. If a cell belongs to multiple clock domains, it is forced to belong to the clock domain with the fastest clock. If a cell belongs to no clock domains, it is forced to belong to the fastest clock domain in the design.

Rail and Supply Nets Based Reports

When you specify the *-rails* option, only the reports associated with the specified rails or supply nets are generated. The *-rails* option accepts a list of name strings. Each name string can contain one or more rail or power supply net names separated by space. One report is generated for each name string if it contains valid rails or supply nets. If the name string contains "all" keyword, a report of all rails is generated. Multiple reports can be generated by using the *report_threshold_voltage_group* command with different combinations of rail names in the list.

Examples

The following script assigns one library to a threshold voltage group named SVT and another library to a group named LFT. After that, the *report_threshold_voltage_group* command reports the threshold voltage groups in a specified order of priority.

```
define_user_attribute -type string -class lib
 default_threshold_voltage_group
set_user_attribute [get_libs agl90g_od_svt_ss]
 default_threshold_voltage_group SVT
set_user_attribute [get_libs agl90g_od_lvt_ss]
 default_threshold_voltage_group LVT

pt_shell> report_threshold_voltage_group -lvth_groups {LVT SVT}
--------------------------------------------------------------------------------
                        Threshold Voltage Group Report

Attributes :
------------
         u -> user-defined power group
         L -> user-defined low Vth group

Power Group          ABCD          LVT(L)        XYZK(L)      SVT
Name                 cell (%)      cell (%)      cell (%)     cell (%)  Attrs
--------------------------------------------------------------------------------
---
io_pad               0 (0.00%)     0 (0.00%)     0 (0.00%)    0 (0.00%)
memory               0 (0.00%)     0 (0.00%)     0 (0.00%)    0 (0.00%)
clock_network        0 (0.00%)     0 (0.00%)     0 (0.00%)    0 (0.00%)
black_box            0 (0.00%)     0 (0.00%)     0 (0.00%)    0 (0.00%)
register             0 (0.00%)     0 (0.00%)     1 (25.00%)   3 (75.00%)
combinational        1 (3.33%)    13 (43.33%)    0 (0.00%)   16 (53.33%)
sequential           0 (0.00%)     4 (100.00%)   0 (0.00%)    0 (0.00%)
--------------------------------------------------------------------------------
---
```

```
Total               1 (2.63%)   17 (44.74%)   1 (2.63%)  19 (50.00%)
-------------------------------------------------------------------------
---
SUMMARY :

          Low Vth cell (%) = 18 (47.37%)
          Total cell (%) =   38 (100.00%)
-------------------------------------------------------------------------
---

1
```

The following example shows reports for rail VDD_MINI4 and four_mini_inst/VDD_MINI4_SWITCH, and for all rails.

```
pt_shell> report_threshold_voltage_group -lvth_groups {LVT} -area -rails
 {"VDD_MINI4 four_mini_inst/VDD_MINI4_SWITCH" "all"}
Report for Rails : VDD_MINI4 four_mini_inst/VDD_MINI4_SWITCH
-------------------------------------------------------------------------
---------------------------
Other Vth:     Cells with a threshold_voltage_group attribute value other
 than
               what has been defined as Low Vth with the -lvth option.
Undefined Vth: Cells which do not have the threshold_voltage_group
 attribute.
-------------------------------------------------------------------------
---------------------------
Attributes :
------------
     u -> user defined power group
     L -> user defined low vth group

Power Group            LVT(L)              DEFAULT_VTH_GROUP   Total
Name                   area (%)            area (%)            area (%)
          Attrs
-------------------------------------------------------------------------
---------------------------
memory             0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
clock_network      0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
black_box          0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
io_pad             0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
combinational      0.000 (  0.00%)     36.176 (100.00%)    36.176
 (100.00%)
register           0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
sequential         29.635 (100.00%)    0.000 (  0.00%)     29.635
 (100.00%)
-------------------------------------------------------------------------
---------------------------
```

```
Total              29.635 ( 45.03%)    36.176 ( 54.97%)    65.811
 (100.00%)
-------------------------------------------------------------------------
--------------------------
SUMMARY :

    Low Vth area (%)       = 29.635 ( 45.03%)
    Other Vth area (%)     = 0.000 (  0.00%)
    Undefined Vth area (%) = 36.176 ( 54.97%)
    -----------------------------------------
    Total area (%)         = 65.811 (100.00%)

-------------------------------------------------------------------------
--------------------------

Report for Rails : all
-------------------------------------------------------------------------
--------------------------
Other Vth:     Cells with a threshold_voltage_group attribute value other
 than
               what has been defined as Low Vth with the -lvth option.
Undefined Vth: Cells which do not have the threshold_voltage_group
 attribute.
-------------------------------------------------------------------------
--------------------------
Attributes :
------------
     u -> user defined power group
     L -> user defined low vth group

Power Group            LVT(L)              DEFAULT_VTH_GROUP   Total
Name                   area (%)            area (%)            area (%)
          Attrs
-------------------------------------------------------------------------
--------------------------
memory             0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
clock_network      0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
black_box          0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
io_pad             0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
combinational      0.000 (  0.00%)     87.170 (100.00%)    87.170
 (100.00%)
register           0.000 (  0.00%)     0.000 (  0.00%)     0.000 (
  0.00%)
sequential         49.392 ( 76.92%)    14.818 ( 23.08%)    64.210
 (100.00%)
-------------------------------------------------------------------------
--------------------------
Total              49.392 ( 32.63%)    101.987 ( 67.37%)   151.379
 (100.00%)
```

```
--------------------------------------------------------------------------------
--------------------------
SUMMARY :

    Low Vth area (%)        = 49.392 ( 32.63%)
    Other Vth area (%)      = 0.000 (  0.00%)
    Undefined Vth area (%) = 101.987 ( 67.37%)
    ------------------------------------------------
    Total area (%)          = 151.379 (100.00%)

--------------------------------------------------------------------------------
--------------------------

1
```

See Also

- define_user_attribute
- report_power
- set_user_attribute
- power_enable_analysis

---

report_timing

Reports the timing paths in the design that have the worst slack.

Syntax

string *report_timing*

```
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-exclude exclude_list
   | -rise_exclude rise_exclude_list
   | -fall_exclude fall_exclude_list]
[-delay_type delay_type]
[-nworst paths_per_endpoint]
[-max_paths max_path_count]
[-group path_group_list]
[-unique_pins]
[-slack_greater_than minimum_slack]
[-slack_lesser_than  maximum_slack]
```

```
[-ignore_register_feedback feedback_slack_cutoff]
[-report_ignored_register_feedback]
[-include_hierarchical_pins]
[-trace_latch_borrow]
[-trace_latch_forward]
[-pba_mode none | path | exhaustive | ml_exhaustive]
[-start_end_type from_to_type]
[-normalized_slack]
[-start_end_pair]
[-cover_design]
[-cover_through through_list]
[-dont_merge_duplicates]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-path_type format]
[-input_pins]
[-nets]
[-nosplit]
[-transition_time]
[-capacitance]
[-significant_digits digits]
[-crosstalk_delta]
[-derate]
[-variation]
[-exceptions dominant | overridden | all]
[-voltage]
[-supply_net_group]
[-physical]
[-sort_by slack | group]
[-tag_paths_filtered_by_pba tag_name]
[-imsa_session session_name]
[-sms_scenarios sms_scenarios_list]
[-domain_crossing domain_crossing_mode]
[-pocv_pruning]
[-attributes attribute_list]
[-timing_path_attributes attribute_list]
[-vdd_slack]
[-vdd_slack_lesser_than maximum_vdd_slack]
[-vdd_slack_greater_than minimum_vdd_slack]
[path_collection]
```

Data Types

from_list	list
rise_from_list	list
fall_from_list	list
to_list	list
rise_to_list	list
fall_to_list	list
through_list	list
rise_through_list	list
fall_through_list	list
exclude_list	list

```
rise_exclude_list          list
fall_exclude_list          list
delay_type                 string
paths_per_endpoint         integer
max_path_count             integer
path_group_list            list
minimum_slack              float
maximum_slack              float
maximum_vdd_slack          float
minimum_vdd_slack          float
feedback_slack_cutoff      float
from_to_type               string
pre_command_string         string
post_command_string        string
format                     string
attribute_list             list
digits                     integer
tag_name                   string
session_name               string
path_collection            collection
sms_scenarios_list         collection
domain_crossing_mode       string
```

Arguments

`-from` *from_list*

Reports only paths that start from the specified list of pins, ports, nets, or cell instances; or from startpoints clocked by the named clocks. Valid startpoints are clock pins of sequential devices and input ports of the design.

`-rise_from` *rise_from_list*

Same as the *-from* option, except that the path must start with a rising transition. If you specify a clock object, the option selects startpoints clocked by the named clock and launched by a rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_from` *fall_from_list*

Same as the *-rise_from* option, except that the path must start with a falling transition.

`-to` *to_list*

Reports only paths that end at the specified list of pins, ports, nets, or cell instances; or at endpoints clocked by the named clocks. Valid endpoints are data input pins of sequential devices and output ports of the design.

Due to the infrastructure of the timing analyzer, the *-to* option runs more efficiently than the *-from* option. As a result, endpoint-oriented reporting flows (based on using the *-to* option) outperform equivalent startpoint-oriented reporting flows (based on using the *-from* option).

`-rise_to` *`rise_to_list`*

Same as the *-to* option, except that the path data must end with a rising transition. If you specify a clock object, the option selects endpoints clocked by the named clock and captured by a rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_to` *`fall_to_list`*

Same as the *-rise_to* option, except that the path data must end with a falling transition.

`-through` *`through_list`*

Reports only paths in which the data goes through the named pins, ports, cell instances, or nets. You can use this option multiple times in a command.

When a single *-through* list contains multiple objects, the data path can go through any one of the listed objects. When you use multiple *-through* options, the data path must meet each of the *-through* options in sequence.

For example, the following command reports a path that starts at A1, then passes through either B1 or B2, then passes through C1, and ends at D1.

```
pt_shell> report_timing -from A1 -through {B1 B2} -through C1 -to
 D1
```

If advanced analysis through transparent latches is enabled (*timing_enable_through_paths* set to *true*), and the pins specified in the last *-through* option are latch loop breakers, then the timing paths reported are constrained by the worst of the closing edges at the latch or by a required value computed using logic downstream of the latch loop breaker. The latter constraint is called "time required through endpoint" in the timing report, and the latch is said to be constrained by "downstream required."

`-rise_through` *`rise_through_list`*

Same as the *-through* option, except that it applies only to paths with a rising transition at the specified objects.

`-fall_through` *`fall_through_list`*

Same as the *-through* option, except that it applies only to paths with a falling transition at the specified objects.

`-exclude` *`exclude_list`*

Excludes all paths in which the data goes from, through, or to the specified list of pins, ports, nets, or cell instances; or excludes all paths belonging to the specified list of path tag sets. This option checks only data paths (not clock paths) for excluded objects.

Feedback

Using the *-exclude* option can result in significantly longer runtimes. To reduce the runtime impact, use this option together with other topological restriction options, such as *-from* and *-through*.

The *-exclude* option does not check borrowing paths considered by the *-trace_latch_borrow* option.

`-rise_exclude` *`rise_exclude_list`*

Same as the *-exclude* option, but applies only to rising transitions at the named pins, ports, nets, or cell instances.

`-fall_exclude` *`fall_exclude_list`*

Same as the *-exclude* option, but applies only to falling transitions at the named pins, ports, nets, or cell instances.

You can use no more than one of the *-exclude*, *-rise_exclude*, or *-fall_exclude* options in a command.

`-delay_type` *`delay_type`*

Specifies the type of path delay constraint to consider for finding and sorting paths with the worst slack:

- *max* (default) -- max delay (setup constraint)
- *min* -- min delay (hold constraint)
- *min_max* -- both min and max delay (reports min first, max second)
- *max_rise* -- max delay, rising at data path endpoint
- *max_fall* -- max delay, falling at data path endpoint
- *min_rise* -- min delay, rising at data path endpoint
- *min_fall* -- min delay, falling at data path endpoint

`-nworst` *`paths_per_endpoint`*

Reports up to the specified number of worst paths per endpoint. The default is 1. A larger value results in a larger report and more runtime. Allowed values are 1 to 2000000.

If you set *-nworst* to any number greater than 1, the command automatically does the following:

- Implicitly sets *-max_paths* equal to the *-nworst* setting if the *-max_paths* option is not used in the command, so that at least one set of multiple paths to an endpoint can be reported.
- Implicitly sets *-slack_lesser_than 0.0* if the *-slack_lesser_than* and *-slack_greater_than* options are not specified, so that only violators are reported.

`-max_paths` *`max_path_count`*

Reports up to the specified maximum total number of paths.

If the *-group* option is specified, the total number of paths reported is at most the specified number of paths per path group.

If the *-group* option is not specified, the total number of paths reported is at most the specified number of paths among all path groups.

The *max_path_count* must be greater than zero. The default *max_path_count* is equal to the *-nworst* setting, or 1 if the *-nworst* option is not specified.

If you set *-max_paths* to a value greater than 1, the command implicitly sets *-slack_lesser_than 0.0* if the *-slack_lesser_than* and *-slack_greater_than* options are not specified, so that only violators are reported.

The *-nworst* option controls the maximum number of paths reported *per endpoint*, whereas the *-max_paths* option controls the *overall total maximum number* of paths reported. The *-max_paths* setting should be at least as large as the *-nworst* setting, so by default the tool implicitly sets *-max_paths* to the same value as *-nworst* if the *-max_paths* option is not specified.

`-group` *`group_name`*

Reports only paths that belong to the specified path groups.

Path groups are created by the *create_clock* and *group_path* commands. Without the *-group* option, the *report_timing* command reports paths without considering path groups. For example, the *report_timing* command alone reports the single worst path in the design, whereas *report_timing -group [get_path_groups]* reports the single worst path *in each path group*.

If the *-sort_by group* option is specified and *-group* is not specified, then an implicit *-group* * specification is used.

`-unique_pins`

Reports only the single worst timing path through any given sequence of pins. No other paths are reported for the same sequence of pins from startpoint to

endpoint. For example, if the worst path starts with a rising edge at the first pin of a pin sequence, then paths starting with a falling edge are not reported for that sequence of pins.

For non-unate logic such as XOR gates, using this option greatly reduces the number of paths reported because of the large number of possible rising/falling edge combinations through each sequence of pins.

Using this option can require longer runtimes when used with the *-nworst* option because many paths must be analyzed to find the worst path through each pin sequence, but only the worst path is reported and counted toward the total number of requested paths.

`-slack_greater_than` *`minimum_slack`*

Reports only paths with slack greater than the specified *minimum_slack* value; these paths have a negative slack better than the specified *minimum_slack* value (or a positive slack that is farther from causing a violation). This option is intended to be used with the *-slack_lesser_than* option to report paths within a specific range of slack.

Unlike the *-slack_lesser_than* option, the *-slack_greater_than* option acts as a post-processing filter to restrict the paths that are reported. As a result, the number of paths reported might be less than the number specified with the *-nworst* and *-max_paths* options.

`-slack_lesser_than` *`maximum_slack`*

Reports only paths with slack less than the specified *maximum_slack* value; these paths have a negative slack worse than the specified *maximum_slack* value (or a positive slack that is closer to causing a violation).

If this option is not specified, the default is computed as follows, highest precedence first:

- If exhaustive PBA is used (*-pba_mode* set to *exhaustive* or *ml_exhaustive*), then the default is 0.0, so that only violators are reported.
- Otherwise, if either *-max_paths* or *-nworst* is set to a value greater than 1, then the default is 0.0, so that only violators are reported.
- Otherwise, the default is "infinity", so that the worst path is reported.

The *-slack_lesser_than* and *-slack_greater_than* options act as filters to restrict the paths that are reported, so the number of paths reported can be fewer than the number specified with the *-nworst* and *-max_paths* options. If no paths meet the slack criteria, no paths are reported.

`-ignore_register_feedback` *`feedback_slack_cutoff`*

Ignores noninverting timing loops that start and end at the same register pin that holds a value. To be ignored, the data-to-output arc and the output-to-data path must be either both inverting or both noninverting. This option applies to minimum delay as well as maximum delay constraints. Paths are ignored only if they have a slack less than the specified *feedback_slack_cutoff* value.

This option acts as a filter to restrict the paths that are reported, so the number of paths reported might be less than the number specified with the *-nworst* and *-max_paths* options.

`-report_ignored_register_feedback`

Displays a list of ignored paths when you use the *-ignore_register_feedback* option.

`-include_hierarchical_pins`

Causes the timing path report to show hierarchical pins as well as leaf pins. The hierarchical pins are shown for informational purposes only; they are reported as having zero incremental delay and they do not affect the timing results.

`-trace_latch_borrow`

Controls the type of report generated for paths that starts at a transparent latch.

If the path startpoint borrows from the previous path segment, using this option causes the report to show the set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop. Each path segment is reported separately, showing the time borrowed and lent and the endpoints of the path segment.

Without the *-trace_latch_borrow* option, the command does not include upstream borrowing latches when in default latch analysis mode or borrowing latch loop breakers when in advanced latch analysis mode (*timing_enable_through_paths* set to *true*) as path segments in the timing report.

`-trace_latch_forward`

Controls the type of report generated for a path that ends at a transparent latch D pin.

If advanced analysis through transparent latches is enabled (*timing_enable_through_paths* set to *true*) and this transparent latch D pin is constrained by a downstream required time, this option causes the report to show the paths in the fanout of this D pin that led to the downstream required constraint. These paths can include multiple path segments whereby the constraint is the result of propagating through more than one latch constrained by the downstream required time. In this case, each path segment is reported separately, showing the downstream required time for that path segment.

Feedback

Without the *-trace_latch_forward* option, the command does not trace the path beyond the specified endpoint.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Specifies one of the following path-based timing analysis modes:

- *none* (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- *path* - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths. To control the ordering of the paths (either by original graph-based slack or recalculated path-based slack), set the *pba_path_mode_sort_by_gba_slack* variable.
- *exhaustive* - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode. You cannot use the *exhaustive* mode together with the *-start_end_pair*, *-cover_design*, or *path_collection* options.
- *ml_exhaustive* - Performs Machine Learning based exhaustive path-based analysis. This mechanism deploys machine learning techniques to trade runtime versus accuracy during the analysis. Accuracy and runtime will approach *-pba_mode exhaustive* as the design approaches zero-slack signoff.

In ordinary graph-based (default) timing analysis, the tool considers both the worst arrival time and worst slew among all signals feeding into a path, even when the worst arrival and worst slew come from different signals.

In path-based timing analysis, the tool considers each path in isolation from other paths, which eliminates impossible combinations of worst slew and worst arrival, and similar combinations of effects such as crosstalk and CRPR. As a result, path-based analysis reduces pessimism and increases accuracy at the cost of more runtime.

`-start_end_type` *`from_to_type`*

Restricts the report to one of four classes of paths based on the type of startpoint and endpoint:

- *reg_to_reg* - from register to register
- *reg_to_out* - from register to output port
- *in_to_reg* - from input port to register
- *in_to_out* - from input port to output port

In this description, "register" means any valid startpoint or endpoint that is not an input port, output port, or bidirectional port, even if not a sequential register. For example, a clock-gating check endpoint qualifies as a "register" in the *in_to_reg* from-to type. A bidirectional port qualifies as both an input port and an output port.

`-normalized_slack`

Gathers, sorts, and reports paths using normalized slack instead of absolute slack. Normalized slack is the absolute slack divided by an idealized maximum allowable propagation delay (typically the clock period). To use this option, the *timing_enable_normalized_slack* variable must be set to *true* for the current timing update.

`-start_end_pair`

Reports the worst path for each startpoint-endpoint pair in the design.

If *-slack_lesser_than* is not specified, all violating startpoint-endpoint pairs are reported (an implicit *-slack_lesser_than* value of 0.0). If *-slack_lesser_than* is specified, the violating startpoint-endpoint pairs meeting that requirement are reported. The number of paths reported is limited to 2000000 unless *-max_paths* is used to exceed this.

The following options can be used to control the *-start_end_pair* behavior:

-max_paths
-nworst
-slack_lesser_than
-from / -rise_from / -fall_from
-through / -rise_through / -fall_through
-to / -rise_to / -fall_to
-exclude / -rise_exclude / -fall_exclude

The *-start_end_pair* option can result in very large reports and long runtimes. Use the preceding options thoughtfully to generate the report. For example, you could use the *-slack_lesser_than* option to specify a slack limit that is only slightly greater than the worst slack in the design.

The following options cannot be used with the *-start_end_pair* option:

-cover_design
-slack_greater_than
-unique_pins
-ignore_register_feedback / -report_ignored_register_feedback
-pba_mode exhaustive | ml_exhaustive
path_collection

The *-start_end_pair* option sorts the paths first by endpoint and then by slack, with the paths sharing the same endpoint reported together in order of increasing slack.

`-cover_design`

Reports the worst path through each violating pin in the design, so that the reported paths cover every violating pin in the design. A pin is deemed to be violating if its slack is less than zero, or less than the value specified with the *-slack_lesser_than* option if that option is used. The slack of a pin is the worst slack among all paths that start at, pass through, or end at that pin. There is no limit on the number of reported paths.

The following options can be used to control the *-cover_design* behavior:

-slack_lesser_than
-from / -rise_from / -fall_from
-through / -rise_through / -fall_through
-to / -rise_to / -fall_to
-exclude / -rise_exclude / -fall_exclude

The following options cannot be used with the *-cover_design* option:

-nworst
-max_paths
-start_end_pair
-unique_pins
-ignore_register_feedback / -report_ignored_register_feedback
-pba_mode exhaustive | ml_exhaustive
path_collection

`-cover_through` *through_list*

Reports the single worst violating path through each of the named pins, ports, cell instances, and nets. Only paths with negative slack are reported (or with slack less than the value specified by the *-slack_lesser_than* option, if used).

The number of paths reported is less than or equal to the number of objects in the *through_list*. Fewer paths are reported if there are no violating paths through the objects, or if the worst path through one object is the same as the worst path through another object; duplicate paths are combined into a single path.

The following options cannot be used with the *-cover_through* option:

-nworst
-max_paths
-start_end_pair
-unique_pins
-through / -rise_through / -fall_through
-exclude / -rise_exclude / -fall_exclude
-ignore_register_feedback / -report_ignored_register_feedback

-pba_mode exhaustive | ml_exhaustive
path_collection

-dont_merge_duplicates

Prevents the merging of duplicate paths across multiple scenarios in distributed multi-scenario analysis (DMSA).

By default, when the same path is reported in more than one scenario, the tool reports only the single most critical instance of that path in the merged report and shows its associated scenario.

If you use this option, the tool does not merge duplicate instances of the same path, but instead reports all critical instances of the path from all scenarios. Because the number of paths reported might be limited by *-nworst*, *-max_paths*, or other command options, the resulting merged report might be more focused on a portion of the design that is critical in multiple scenarios, and less evenly spread out across different portions of design.

-pre_commands *pre_command_string*

Specifies a list of commands, separated by semicolons, to be executed in the worker context before execution of the *report_timing* command. This option is available only if you invoke the PrimeTime tool with the *-multi_scenario* option. The maximum size of a command is 1000 characters.

-post_commands *post_command_string*

This option is like the *-pre_commands* option, except that the commands are executed after (instead of before) the *report_timing* command.

-path_type *format*

Specifies the types of path information displayed in the timing report, using one of the following format keywords:

- *summary* -- Displays the path startpoint, path endpoint, and slack.
- *end* -- Displays the path endpoint, path delay, required time, and slack.
- *short* -- Displays a report like the default *full* report but omits the intermediate points between the startpoint and endpoint.
- *full* (the default) -- Displays the full data path, including all intermediate points and their incremental and cumulative delay, together with the launch and capture clock arrival times, data required time, and slack.
- *full_clock* -- Displays a *full* report with the addition of the clock path points from the clock source to the launch and capture points.

- *full_clock_expanded* -- Displays a *full_clock* report with the addition of the clock path points between the primary clock source and its related generated clock source point. This report is different from a *full_clock* report only when the clock is a generated clock.

`-input_pins`

Shows cell input pins as well as cell output pins in the timing path. As a result, the report shows the incremental net and cell delays separately at each point, instead of combined. By default, the report shows only the cell output pins.

`-nets`

Shows nets in the timing path. By default, the report does not show nets.

`-nosplit`

Prevents line splitting. This can useful for scripts that extract information from the report. By default, the report generates a new line when the text cannot fit in the alloted space in a column.

`-transition_time`

Shows the transition time (slew) in the path report for each driver pin and load pin, appearing as an additional column labeled "Trans". By default, the report does not show transition time.

`-capacitance`

Shows the total capacitance in the path report for each net, appearing as an additional column labeled "Cap". By default, the report does not show capacitance.

The reported values consider either the CCS receiver capacitance (the default) or the lumped capacitance for pins, depending on the *report_capacitance_use_ccs_receiver_model* variable setting.

`-significant_digits` *digits*

Specifies the number of digits after the decimal point displayed for time values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which is 2 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

`-crosstalk_delta`

Reports the annotated delta delay values at cell input pins in a column labeled "Delta". The *-input_pins* option is automatically selected. The delta values are computed during crosstalk signal integrity analysis, or they can be annotated manually by using the *set_annotated_delay -delta_only* and *set_annotated_transition -delta_only* commands. Note that the *-crosstalk_delta*

option only reports previously calculated or annotated delta values; it does not initiate crosstalk analysis. Delta transition times are also shown if you use the *-transition_time* option.

`-derate`

Shows derating factors in a column labeled "Derate". The default is to not show derating factors. Specifying this option automatically sets the *-input_pins* option, so that different net and cell derating values are reported. When you use the *-path_type full_clock_expanded* option, an additional summary report follows the timing report showing the overall effect of derating the path. Derating factors are always shown with at least two significant digits.

`-variation`

Includes parametric on-chip variation (POCV) information in the report in columns labeled "Mean" and "Sensit". POCV analysis is enabled by setting the *timing_pocvm_enable_analysis* variable to *true*.

`-exceptions dominant | overridden | all`

Reports user-specified timing exceptions that are satisfied per timing path being reported.

Specifying *dominant* reports the dominant timing exception in the path. It can be one of the following: false path, minimum and maximum delays, and multicycle paths.

Specifying *overridden* reports all the timing exceptions that were overridden by the dominant timing exception.

The unconstrained paths and their reasons are reported for all three options if *timing_report_unconstrained_paths* variable is set to *true*, otherwise unconstrained paths are not reported.

To get more information about the exceptions that apply to the path, you can create a path collection with the *get_timing_paths* command, then query certain timing path attributes. For details, see the man page for the *get_timing_paths* command.

Note: The additional per-path analysis required by the *-exceptions* option can be significant.

This option cannot be used with the *-path_type short/end/summary* option.

`-voltage`

Reports the operating voltage for each path element in a column labeled "Voltage", allowing you to debug voltage levels in multivoltage designs. The displayed voltage is the voltage used in delay calculation, which is typically the

voltage of the operating condition, related supply net, related power pin, or pin signal.

`-supply_net_group`

Reports the supply group or the supply net name for each path element in a column labeled "Supply-Net-Group". If the supply net group name is not defined, the column shows the word "unconnected". If the supply group is not defined and the supply net name is available, then the supply net name is reported in the column along with the letter "n".

`-physical`

Reports the X-Y coordinates for each path element in a column labeled "Location". The location information must be available from previous usage of the *read_parasitics* command with the *read_parasitics_load_locations* variable set to *true*. If the X-Y coordinate data is not available for some or all of the elements, the report shows the string "unplaced" for the missing data.

`-sort_by slack | group`

Specifies the sorting of the reported paths, either by *slack* or by *group*. Ordering by slack means ordering the path reports strictly by slack, ignoring group membership. Sorting by group means ordering the path reports by group, and within each group, by slack.

The default is to sort by *slack*. However, if *-group* is specified, then the default is to sort by *group*.

`-tag_paths_filtered_by_pba` *`tag_name`*

When used with the *-pba_mode path* option, the *-tag_paths_filtered_by_pba* option causes tagging of all paths originally found by path-based analysis and later discarded by path-based analysis, using the specified tag name. In future *get_timing_paths* and *report_timing* commands, you can exclude the tagged paths from further analysis by specifying the tag name as an argument to the *-exclude* option.

`-imsa_session` *`session_name`*

This option can only be used in IMSA mode, when the *eco_update_path_timing* variable is set to *true*. By default, *report_timing* in IMSA mode covers all in-memory timing path collections. When this option is used, *report_timing* covers the timing paths from the specified session only.

`path_collection`

Specifies a collection of timing paths to report, for example, a collection previously created by *set mypaths [get_timing_paths ...]*. When you use this option with the *-pba_mode path* option, the command performs path-based analysis on the paths in the *path_collection* before they are reported. This option

is mutually exclusive with options that control the selection of paths to report; it is compatible only with options that control the formatting of the report.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only timing paths compatible to the specified SMS scenarios collection are reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

`-domain_crossing` *`domain_crossing_mode`*

Specifies the domain crossing report mode. Allowed values include *all, only_crossing, exclude_crossing*. The default mode is *all*. This option is applied as a reporting filter, ensuring that only paths which satisfy the specified mode are reported. With the default setting of *all*, all paths are considered for reporting, including cross-domain and intra-domain. With the *only_crossing* mode, reports include only cross-domain paths. Using the *exclude_crossing* mode only reports intra-domain paths.

This option is only available with SMVA or SMC analysis.

`-pocv_pruning`

Generates a path collection that contributes to joint success rate. This option prunes away all subcritical paths that have no impact on high sigma failure rate. The returned path collection can be used with the *get_design_variation* command for design variation analysis.

This option requires that the *-pba_mode exhaustive* option also be used. It can significantly reduce runtime and memory of both *get_timing_paths* and *get_design_variation*, so it is recommended for use when generating path collections for design variation analysis.

`-attributes` *`attribute_list`*

Specifies a list of *timing_point* attributes to be printed inline with each timing point row.

`-timing_path_attributes` *`attribute_list`*

Specifies a list of *timing_path* attributes to be printed in the report header of each path (that includes information such as startpoint, endpoint, and the last common pin).

`-vdd_slack`

Shows voltage slack and voltage sensitivity information in the report for path collections generated with *get_timing_paths -vdd_slack_lesser_than* or *get_timing_paths -vdd_slack_greater_than*. The *-vdd_slack* option must be used together with a *path_collection* argument.

`-vdd_slack_lesser_than` *`maximum_vdd_slack`*

Performs voltage slack analysis on the paths and shows voltage slack in the report if the value is less than the specified *maximum_vdd_slack value.*

The following options must also be used with the *-vdd_slack_lesser_than* option:

```
-pba_mode path | exhaustive
-path_type full_clock_expanded
```

`-vdd_slack_greater_than` *`minimum_vdd_slack`*

Performs voltage slack analysis on the paths and shows voltage slack in the report if the value is greater than the specified *minimum_vdd_slack value.*

The following options must also be used with the *-vdd_slack_greater_than* option:

```
-pba_mode path | exhaustive
-path_type full_clock_expanded
```

Description

The *report_timing* command reports the timing paths in the current design that have the worst slack. These are the paths that violate the timing constraints by the largest amounts, or paths with positive slack that come closest to causing a timing violation.

Each path has a startpoint and an endpoint. Data is launched by a clock edge at the path startpoint, propagated through combinational logic in the path, and then captured at the path endpoint by another clock edge. The startpoint can be a register clock pin or an input port. The endpoint can be a register data input pin or an output port.

To restrict the scope of the report to only the paths that start, pass through, or end on specific pins, ports, nets, or cell instances, use the *-from*, *-though*, and *-to* options, and the similar transition-specific options (*-rise_from*, *-fall_to*, and so on). To restrict the report to specific clocks, you can use the same options (for example, *-to [get_clocks CLK1]*) or the *-group* option.

By default, the *report_timing* command without options reports the single path in the design with the worst max delay (setup constraint) violation. To consider constraints other than max delay, use the *-delay_type* option. To control the number of paths reported, use the *-nworst* option, which specifies the maximum number of worst paths reported per endpoint; and the *-max_paths* option, which specifies the overall maximum number of paths reported by the command.

To report paths whose slack values fall within a specific range, use the *-slack_lesser_than* and *-slack_greater_than* options in combination with the *-nworst* and *-max_paths* options.

The default timing report shows the point-to-point sequence of cell output pins in the data path, as in the following example:

```
Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
 Endpoint: pad[1] (output port clocked by PCI_CLK)
 Path Group: PCI_CLK
 Path Type: max

 Point                                                  Incr       Path
 -----------------------------------------------------------------------
 clock PCI_CLK (rise edge)                             0.000      0.000
 clock network delay (propagated)                      1.105      1.105
 I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)          0.000      1.105 r
 I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)           0.404 &    1.509 r
 U7/ZN (inv0d1)                                        0.066 &    1.575 f
 U62/ZN (inv0d1)                                       0.042 &    1.617 r
 U63/Z (or02d7)                                        0.269 &    1.887 r
 U63ASTipoInst495/Z (bufbdk)                           0.306 &    2.192 r
 U63ASTipoInst494/Z (bufbda)                           3.798 &    5.991 r
 pad_iopad_1/PAD (pc3b03)                              6.118 H   12.109 r
 pad[1] (inout)                                        0.048     12.157 r
 data arrival time                                               12.157

 clock PCI_CLK (rise edge)                            15.000     15.000
 clock network delay (propagated)                      0.000     15.000
 output external delay                                -4.000     11.000
 data required time                                              11.000
 -----------------------------------------------------------------------
 data required time                                              11.000
 data arrival time                                              -12.157
 -----------------------------------------------------------------------
 slack (VIOLATED)                                                -1.157
```

The report shows the incremental delay contributed by each pin-to-pin segment of the path (Incr column) and the cumulative path delay up to each pin in the timing path (Path column). The table also shows the data launch clock arrival time, the data capture clock arrival time, an accounting of arrival time versus required time, and the resulting timing slack for the path. A negative slack indicates a timing violation.

To calculate the delay contributed by each point along the path, the tool considers any back-annotated RC network or SDF delay information. The types of back-annotated data on path elements in the timing path are indicated by a character symbol in the Incr column:

```
Symbol      Annotation
------      ----------
  H         Hybrid annotation
  ^         Ideal network latency annotation
  *         SDF back-annotation
  &         RC network back-annotation
  $         RC pi back-annotation
  +         Lumped RC
```

```
  @          HyperScale annotation
none         Wire-load model or none
```

Certain annotations dominate others. For example, SDF takes precedence over back-annotated RC parasitics. The symbol displayed in the Incr column indicates the dominant annotation on the path element.

If the *-input_pins* option is used, the report includes the input pins as well as output pins of each cell in the path. In that case, each pin-to-pin delay spans either a net or cell. Without the *-input_pins* option, the delay shown can span both a net and a cell. If the net and cell have the same dominant annotation, the appropriate symbol is shown; otherwise, the "H" symbol indicates that a hybrid of annotation types exists on the corresponding path segment.

The dominant annotation is not necessarily the actual annotation used to calculate the delay. For example, suppose that a back-annotated RC network exists on a net, but the network calculation fails to converge for the driver cell delay. In that case, you get a warning message and a lumped RC model is used instead. However, in the timing report, the "&" symbol still reports the RC network annotation as the dominant annotation. To see the actual pin-to-pin delay calculation in detail, use the *report_delay_calculation* command.

To control the amount of detail provided in the point-to-point timing report, use the following options:

- *-path_type* -- Specifies the report format: *short*, *full*, *full_clock*, ...
- *-input_pins* -- Lists the cell input pins (in addition to output pins) in the path
- *-nets* -- Lists the nets (as well as pins) in the path
- *-transition_time* -- Shows transition time (slew) in a separate column
- *-capacitance* -- Shows capacitance values in a separate column
- *-crosstalk_delta* -- Shows annotated delta delay values in a separate column
- *-derate* -- Shows derating factors in a separate column
- *-variation* -- Shows POCV mean and sensitivity values in separate columns
- *-voltage* -- Shows operating voltages in a separate column
- *-physical* -- Shows XY coordinates a separate column

To invoke path-based analysis, use the *-pba_mode* option. In path-based timing analysis, the tool considers each path in isolation from other paths, which eliminates impossible combinations of worst slew and worst arrival, and similar combinations of effects such as crosstalk and CRPR. As a result, path-based analysis reduces pessimism and increases accuracy at the cost of more runtime.

Feedback

Examples

The following example reports the single path in the design with the worst max-delay (setup) slack:

```
pt_shell> report_timing
****************************************
Report : timing
        -path_type full
        -delay_type max
        -max_paths 1
        -sort_by slack
        ...
****************************************

  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[1] (output port clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max
  Min Clock Paths Derating Factor : 0.900

  Point                                                    Incr       Path
  -----------------------------------------------------------------------
  clock PCI_CLK (rise edge)                               0.000      0.000
  clock network delay (propagated)                        1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)            0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)             0.404 &    1.509 r
  U7/ZN (inv0d1)                                          0.066 &    1.575 f
  U62/ZN (inv0d1)                                         0.042 &    1.617 r
  U63/Z (or02d7)                                          0.269 &    1.887 r
  U63ASTipoInst495/Z (bufbdk)                             0.306 &    2.192 r
  U63ASTipoInst494/Z (bufbda)                             3.798 &    5.991 r
  pad_iopad_1/PAD (pc3b03)                                6.118 H   12.109 r
  pad[1] (inout)                                          0.048     12.157 r
  data arrival time                                                 12.157

  clock PCI_CLK (rise edge)                              15.000     15.000
  clock network delay (propagated)                        0.000     15.000
  clock reconvergence pessimism                           0.000     15.000
  output external delay                                  -4.000     11.000
  data required time                                                11.000
  -----------------------------------------------------------------------
  data required time                                                11.000
  data arrival time                                                -12.157
  -----------------------------------------------------------------------
  slack (VIOLATED)                                                  -1.157
```

The following example shows the cell input pins as well as cell output pins in the path. As a result, the report shows the incremental net and cell delays separately at each point.

```
pt_shell> report_timing -input_pins
  ...
```

```
  Point                                                  Incr       Path
 ---------------------------------------------------------------------------
  clock PCI_CLK (rise edge)                            0.000      0.000
  clock network delay (propagated)                     1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)         0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)          0.404 &    1.509 r
  U7/I (inv0d1)                                        0.008 &    1.517 r
  U7/ZN (inv0d1)                                       0.058 &    1.575 f
  U62/I (inv0d1)                                       0.004 &    1.579 f
  U62/ZN (inv0d1)                                      0.038 &    1.617 r
  U63/A2 (or02d7)                                      0.004 &    1.621 r
  U63/Z (or02d7)                                       0.265 &    1.887 r
  U63ASTipoInst495/I (bufbdk)                          0.054 &    1.941 r
  U63ASTipoInst495/Z (bufbdk)                          0.252 &    2.192 r
  U63ASTipoInst494/I (bufbda)                          2.664 &    4.857 r
  U63ASTipoInst494/Z (bufbda)                          1.134 &    5.991 r
  pad_iopad_1/OEN (pc3b03)                             1.444 &    7.434 r
  pad_iopad_1/PAD (pc3b03)                             4.675     12.109 r
  pad[1] (inout)                                       0.048     12.157 r
  data arrival time                                              12.157
  ...
```

The following example shows the net names and net fanout as well as cell output pins in the path.

```
pt_shell> report_timing -nets
  ...
  Point                                        Fanout      Incr       Path
 ---------------------------------------------------------------------------
----
  clock PCI_CLK (rise edge)                                0.000      0.000
  clock network delay (propagated)                         1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)             0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)              0.404 &    1.509 r
  I_ORCA_TOP/I_PCI_CORE/pad_en (net)             2
  U7/ZN (inv0d1)                                           0.066 &    1.575 f
  n43 (net)                                      1
  U62/ZN (inv0d1)                                          0.042 &    1.617 r
  n8 (net)                                       1
  U63/Z (or02d7)                                           0.269 &    1.887 r
  n134 (net)                                     8
  U63ASTipoInst495/Z (bufbdk)                              0.306 &    2.192 r
  n134ASTipoNet251 (net)                         3
  U63ASTipoInst494/Z (bufbda)                              3.798 &    5.991 r
  n134ASTipoNet250 (net)                         1
  pad_iopad_1/PAD (pc3b03)                                 6.118 H   12.109 r
  pad[1] (net)                                   1
  pad[1] (inout)                                           0.048     12.157 r
```

```
  data arrival time                                                  12.157
  ...
```

The following example reports the worst path that passes through a specified cell and ends at a specified port. The pins of the "through" cell are marked with arrow notation "<-" in the report.

```
pt_shell > report_timing -through [get_cells U63] -to [get_ports pad[1]]
  ...

  Point                                                 Incr       Path
  ---------------------------------------------------------------------------
  clock PCI_CLK (rise edge)                          0.000      0.000
  clock network delay (propagated)                   1.105      1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)       0.000      1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)        0.404 &    1.509 r
  U7/ZN (inv0d1)                                     0.066 &    1.575 f
  U62/ZN (inv0d1)                                    0.042 &    1.617 r
  U63/A2 (or02d7) <-                                 0.004 &    1.621 r
  U63/Z (or02d7) <-                                  0.265 &    1.887 r
  U63ASTipoInst495/Z (bufbdk)                        0.306 &    2.192 r
  U63ASTipoInst494/Z (bufbda)                        3.798 &    5.991 r
  pad_iopad_1/PAD (pc3b03)                           6.118 H   12.109 r
  pad[1] (inout)                                     0.048     12.157 r
  data arrival time                                            12.157

  clock PCI_CLK (rise edge)                         15.000     15.000
  clock network delay (propagated)                   0.000     15.000
  clock reconvergence pessimism                      0.000     15.000
  output external delay                             -4.000     11.000
  data required time                                           11.000
  ---------------------------------------------------------------------------
  data required time                                           11.000
  data arrival time                                           -12.157
  ---------------------------------------------------------------------------
  slack (VIOLATED)                                             -1.157
```

The following example reports the endpoint path delay, required time, clock reconvergence pessimism removal (if applicable), and slack for the five worst max-delay paths:

```
pt_shell> report_timing -path_type end -max_paths 5
 ...

Endpoint                          Path Delay     Path Required    CRP
 Slack
-------------------------------------------------------------------------------
----
pad[1] (inout)                      12.157 r       11.000        0.000
 -1.157
I_ORCA_TOP/I_BLENDER/s4_op2_reg[31]/D (sdnrb1)
                                    12.004 f       10.547        0.470
 -0.987
```

```
pad[14] (inout)                    11.874 r       11.000        0.000
 -0.874
pad[13] (inout)                    11.874 r       11.000        0.000
 -0.874
I_ORCA_TOP/I_BLENDER/s4_op1_reg[31]/D (sdnrb1)
                                   11.643 f       10.520        0.282
 -0.841
```

The following example includes capacitance and transition time in the report:

```
pt_shell> report_timing -capacitance -transition_time
  ...

  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[1] (output port clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max
  Min Clock Paths Derating Factor : 0.900

  Point                                    Cap      Trans       Incr
 Path

 ---------------------------------------------------------------------------
-----
  clock PCI_CLK (rise edge)                                    0.000
 0.000
  clock network delay (propagated)                             1.105
 1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)
                                                  0.363        0.000
 1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)
                                         0.008    0.140        0.404 &
 1.509 r
  U7/ZN (inv0d1)                         0.004    0.089        0.066 &
 1.575 f
  U62/ZN (inv0d1)                        0.004    0.071        0.042 &
 1.617 r
  U63/Z (or02d7)                         0.060    0.207        0.269 &
 1.887 r
  U63ASTipoInst495/Z (bufbdk)            3.458    0.136        0.306 &
 2.192 r
  U63ASTipoInst494/Z (bufbda)            1.725    0.524        3.798 &
 5.991 r
  pad_iopad_1/PAD (pc3b03)               8.586    1.272        6.118 H
 12.109 r
  pad[1] (inout)                                   1.272        0.048
 12.157 r
  data arrival time
 12.157

  ...
```

Feedback

The following example reports nets and input pins, and includes capacitance and transition time in the report:

```
pt_shell> report_timing -nets -input_pins -capacitance -transition_time
  ...

  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop clocked by PCI_CLK)
  Endpoint: pad[1] (output port clocked by PCI_CLK)
  Path Group: PCI_CLK
  Path Type: max
  Min Clock Paths Derating Factor : 0.900

  Point                        Fanout    Cap     Trans      Incr
 Path

 ---------------------------------------------------------------------------
---
  clock PCI_CLK (rise edge)                                 0.000
 0.000
  clock network delay (propagated)                          1.105
 1.105
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)
                                                 0.363      0.000
 1.105 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)    0.140      0.404 &
 1.509 r
  I_ORCA_TOP/I_PCI_CORE/pad_en (net)
                                2    0.008
  U7/I (inv0d1)                                  0.142      0.008 &
 1.517 r
  U7/ZN (inv0d1)                                 0.089      0.058 &
 1.575 f
  n43 (net)                     1    0.004
  U62/I (inv0d1)                                 0.089      0.004 &
 1.579 f
  U62/ZN (inv0d1)                                0.071      0.038 &
 1.617 r
  n8 (net)                      1    0.004
  U63/A2 (or02d7)                                0.072      0.004 &
 1.621 r
  U63/Z (or02d7)                                 0.207      0.265 &
 1.887 r
  n134 (net)                    8    0.060
  U63ASTipoInst495/I (bufbdk)                    0.276      0.054 &
 1.941 r
  U63ASTipoInst495/Z (bufbdk)                    0.136      0.252 &
 2.192 r
  n134ASTipoNet251 (net)        3    3.458
  U63ASTipoInst494/I (bufbda)                    8.904      2.664 &
 4.857 r
  U63ASTipoInst494/Z (bufbda)                    0.524      1.134 &
 5.991 r
```

```
 n134ASTipoNet250 (net)          1    1.725
 pad_iopad_1/OEN (pc3b03)                      4.964     1.444 &
7.434 r
 pad_iopad_1/PAD (pc3b03)                      1.272     4.675
12.109 r
 pad[1] (net)                    1    8.586
 pad[1] (inout)                                1.272     0.048
12.157 r
 data arrival time
12.157
 ...
```

The following example shows an unconstrained path. The command first looks for constrained paths; if none are found, it reports unconstrained paths.

```
pt_shell> set_app_var timing_report_unconstrained_paths true
true
pt_shell> report_timing -to [get_ports pad[5]]
 ...
  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
              (rising edge-triggered flip-flop)
  Endpoint: pad[5] (output port)
  Path Group: (none)
  Path Type: max

  Point                                                   Incr       Path
  ---------------------------------------------------------------------------
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)          0.000      0.000 r
  I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)           0.404 &    0.404 r
  U7/ZN (inv0d1)                                        0.069 &    0.473 f
  U62/ZN (inv0d1)                                       0.043 &    0.515 r
  U63/Z (or02d7)                                        0.269 &    0.785 r
  U63ASTipoInst499/Z (bufbda)                           0.243 &    1.027 r
  U63ASTipoInst498/Z (bufbda)                           2.205 &    3.232 r
  pad_iopad_5/PAD (pc3b03)                              6.121 H    9.354 r
  pad[5] (inout)                                        0.048      9.402 r
  data arrival time                                                9.402
  ---------------------------------------------------------------------------
  (Path is unconstrained)
```

The following example shows a timing report with the *-derate* and *-path_type full_clock_expanded* options. The *-derate* option shows the applicable derating factors in a separate column. The *-path_type full_clock_expanded* option reports the clock path points from the clock source to the launch and capture points, including points from the primary clock and its related generated clock; and also displays a derating summary report at the end.

```
pt_shell> set_timng_derate -late 1.04
1
pt_shell> report_timing -derate -path_type full_clock_expanded
 ...
  Startpoint: I_ORCA_TOP/I_PCI_CORE/pad_en_reg
```

```
             (rising edge-triggered flip-flop clocked by PCI_CLK)
Endpoint: pad[1] (output port clocked by PCI_CLK)
Path Group: PCI_CLK
Path Type: max

Point                              Derate    Incr       Path
-----------------------------------------------------------
clock PCI_CLK (rise edge)                   0.000      0.000
clock source latency                        0.000      0.000
pclk (in)                                   0.000      0.000 r
pclk_iopad/PAD (pc3d01)            1.040    0.048      0.048 r
pclk_iopad/CIN (pc3d01)            1.040    0.792 &    0.841 r
I_CLOCK_GEN/I_PLL_PCI/REF_CLK (PLL)
                                   1.040    0.034 &    0.874 r
I_CLOCK_GEN/I_PLL_PCI/CLK (PLL)
                                   1.040   -1.211 *   -0.337 r
I_CLOCK_GEN/bufbdfG1B1I1_1/I (bufbdf)
                                   1.040    0.008 &   -0.329 r
I_CLOCK_GEN/bufbdfG1B1I1_1/Z (bufbdf)
                                   1.040    0.205 &   -0.124 r
I_CLOCK_GEN/U21/I0 (mx02d2)        1.040    0.004 &   -0.120 r
I_CLOCK_GEN/U21/Z (mx02d2)         1.040    0.187 &    0.067 r
I_CLOCK_GEN/bufbdfG2B1I1_2/I (bufbdf)
                                   1.040    0.008 &    0.075 r

...

pad_iopad_1/PAD (pc3b03)           1.040    4.862     12.694 r
pad[1] (inout)                     1.040    0.050     12.745 r
data arrival time                                     12.745

clock PCI_CLK (rise edge)                  15.000     15.000
clock network delay (propagated)            0.000     15.000
clock reconvergence pessimism               0.000     15.000
output external delay                      -4.000     11.000
data required time                                    11.000
-----------------------------------------------------------
data required time                                    11.000
data arrival time                                    -12.745
-----------------------------------------------------------
slack (VIOLATED)                                      -1.745

Derate Summary Report
-------------------------------------------------
total derate : required time                0.000
total derate : arrival time                -0.587
-------------------------------------------------
total derate : slack                        0.587

slack (with derating applied) (VIOLATED)
                                           -1.745
clock reconvergence pessimism (due to derating)
                                            0.000
```

```
  --------------------------------------------------
  slack (with no derating) (VIOLATED)    -1.157
```

The following example includes crosstalk delta delays and transition times:

```
pt_shell> report_timing -transition_time -capacitance -nets \\
          -crosstalk_delta -significant_digits 4
****************************************
Report : timing
        -path_type full
        -delay max
        -input_pins
        -nets
        -max_paths 1
        -transition_time
        -capacitance
        -crosstalk_delta
Design : TestBH
****************************************

  Startpoint: FirstReg (rising edge-triggered flip-flop clocked by
 myclock2)
  Endpoint: SecondReg (rising edge-triggered flip-flop clocked by
 myclock3)
  Path Group: myclock3
  Path Type: max

  Point                        Fanout    Cap    DTrans    Trans    Delta
 Incr       Path

 ---------------------------------------------------------------------------
------------------
  clock myclock2 (rise edge)
 2.0000     2.0000
  clock network delay (propagated)
 0.0000     2.0000
  FirstReg/CP (FD1Q)                                  0.1634
 0.0000     2.0000 r
  FirstReg/Q (FD1Q)                                   0.4030
 0.4286 &   2.4286 f
  OutFirstReg (net)              1   0.0996
  InstSimple/InC (SimpleComb)                         0.0000
 0.0000     2.4286 f
  InstSimple/InC (net)
  InstSimple/InterNand/B (ND2)               0.1435   0.5467   0.1030
 0.1092 &   2.5377 f
  InstSimple/InterNand/Z (ND2)                        0.7001
 0.3598 &   2.8976 r
  InstSimple/OutS (net)          1   0.1001
  InstSimple/OutS (SimpleComb)                        0.0000
 0.0000     2.8976 r
  Reg (net)
```

```
  SecondReg/D (FD1Q)                              0.1540   0.8542   0.2562
 0.2615 &   3.1590 r
  data arrival time
          3.1590

  clock myclock3 (rise edge)
 3.0000     3.0000
  clock network delay (propagated)
 0.0000     3.0000
  SecondReg/CP (FD1Q)
          3.0000 r
  library setup time
 -0.2251     2.7749
  data required time
          2.7749

 ------------------------------------------------------------------------
------------------
  data required time
          2.7749
  data arrival time
         -3.1590

 ------------------------------------------------------------------------
------------------
  slack (VIOLATED)
         -0.3841
```

The following example shows a timing report with the *-exceptions all* option:

```
pt_shell> report_timing -from ff1/CP -to ff3/D -exceptions all
****************************************
Report : timing
        -path_type full
        -delay max
        -max_paths 1
 ...
****************************************

  Startpoint: ff1 (rising edge-triggered flip-flop clocked by CLK)
  Endpoint: ff3 (rising edge-triggered flip-flop clocked by CLK)
  Path Group: CLK
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  ff1/CP (FD1)                             0.00       0.00 r
  ff1/Q (FD1)                              1.43       1.43 f
  inv1/Z (IV)                              0.54       1.97 r
  and1/Z (AN2)                             0.80       2.78 r
  inv3/Z (IV)                              0.25       3.02 f
  ff3/D (FD1)                              0.00       3.02 f
  data arrival time                                   3.02
```

```
  max_delay                                3.00       3.00
  library setup time                      -0.80       2.20
  data required time                                  2.20
  ------------------------------------------------------------
  data required time                                  2.20
  data arrival time                                  -3.02
  ------------------------------------------------------------
  slack (VIOLATED)                                   -0.83


The dominant exceptions are:
From          Through        To             Setup                      Hold
------------------------------------------------------------------------
---

*             inv3/Z         *              max=3

The overridden exceptions are:
From          Through        To             Setup                      Hold
------------------------------------------------------------------------
---

*             inv1/Z         *              max=5

1
```

To report the worst path in the design using path-based analysis:

```
pt_shell> report_timing -pba_mode exhaustive
 ...
```

To report all endpoints in the design which fail after considering path-based analysis:

```
pt_shell> report_timing -pba_mode exhaustive -slack_lesser_than 0
 -max_paths 1000
 ...
```

The following example shows a timing report with the *-attributes* option:

```
pt_shell> report_timing -attributes {transition object.net.full_name}
****************************************
Report : timing
 -path_type full
 -delay_type max
 -max_paths 1
****************************************

  Startpoint: hd1/cell8/latches/lln/GN
              (internal path startpoint clocked by CLK')
  Endpoint: hd2/cell1/latches/ffp
              (rising edge-triggered flip-flop clocked by CLK')
  Last common pin: CLK
```

```
  Path Group: CLK
  Path Type: max

  Point                                                    Incr       Path
      transition  object.net

 ---------------------------------------------------------------------------
-----------------------------------------
  clock CLK' (fall edge)                                   0.00       0.00
  clock network delay (propagated)                      1121.61    1121.61
  hd1/cell8/latches/lln/GN (LD2)                           0.00    1121.61
 f          0.01  hd1/cell8/latches/CLK
  hd1/cell8/latches/lln/Q (LD2)                            1.38    1123.00
 f          0.14  hd1/cell8/latches/llnq
  hd1/cell8/latches/xora/Z (EO)                            1.13    1124.12
 f          0.07  hd1/cell8/latches/llnet
  hd1/cell8/latches/xorb/Z (EO)                            1.19    1125.31
 f          0.13  hd1/cell8/latches/Z
  hd1/xor12/Z (EO)                                         1.19    1126.50
 f          0.13  hd1/net78
  hd1/xor13/Z (EO)                                         1.19    1127.69
 f          0.13  hd1/net5678
  hd1/xor14/Z (EO)                                         1.13    1128.82
 f          0.07  hd1/Z
  xor2/Z (EO)                                              7.79    1136.61
 r          7.00  dnet
  hd2/cell1/latches/ffp/D (FD1)                            0.00    1136.61
 r          7.00  hd2/cell1/latches/D
  ...
```

See Also

- get_timing_paths
- report_constraint
- report_delay_calculation
- report_supply_group
- set_case_analysis
- report_default_significant_digits
- si_enable_analysis
- timing_enable_through_paths
- timing_report_always_use_valid_start_end_points
- timing_report_fixed_width_columns_on_left
- timing_report_unconstrained_paths

Feedback

report_timing_budget

Reports the budget allocation specifications applied by the *set_timing_budget* command.

Syntax

string *report_timing_budget*

```
-verbose
```

Data Types

verbose	boolean

Arguments

`-verbose`

Reports all the hierarchical pins of a block separately.

Examples

The following command reports the budget allocation specifications applied to the current design:

```
pt_shell> report_timing_budget

Point                Budget Allocation Mode      Path Type       Value
-------------------------------------------------------------------------
<Global Mode>        slack_margin                                 1.00
BLK1/Z               delay_value                 max              3.00
```

See Also

- set_timing_budget
- update_budget

report_timing_derate

Reports derating previously set by the *set_timing_derate* command.

Syntax

status *report_timing_derate*

```
[-include_inherited]
[-aocvm_guardband] [-pocvm_guardband] [-pocvm_coefficient_scale_factor]
[-pocvm_subtract_sigma_factor_from_nominal]
[-increment]
[-min_period]
[-min_pulse_width]
```

```
[-significant_digits digits]
[-nosplit]
[-sms_scenarios sms_scenarios_list]
[object_list]
```

Data Types

```
digits                int
object_list           list
sms_scenarios_list    collection
```

Arguments

`-include_inherited`

Reports the name of the design, ancestor cell, or library cell from which each object inherits its derating settings.

`-aocvm_guardband`

Reports derating previously set by the *set_timing_derate -aocvm_guardband* command.

`-pocvm_guardband`

Reports derating previously set by the *set_timing_derate -pocvm_guardband* command.

`-pocvm_coefficient_scale_factor`

Reports derating previously set by the *set_timing_derate -pocvm_coefficient_scale_factor* command.

`-pocvm_subtract_sigma_factor_from_nominal`

Reports derating previously set by the *set_timing_derate -pocvm_subtract_sigma_factor_from_nominal* command.

`-increment`

Reports incremental derating adjustments previously set by the *set_timing_derate -increment* command.

`-min_period`

Reports min period constraint derating adjustments previously set by the *set_timing_derate -min_period* command.

`-min_pulse_width`

Reports min pulse width constraint derating adjustments previously set by the *set_timing_derate -min_pulse_width* command.

`-significant_digits` *digits*

Specifies the number of digits after the decimal point to be displayed for values in the report. The default is determined by the *report_default_significant_digits* variable.

`-nosplit`

Prevents splitting of long lines of text in the report.

object_list

Reports derating for the given list or collection of cells, library cells, or nets. To report the names of parent objects from which these objects inherit their derating factors, also use the *-include_inherited* option in the command.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios to analyze. Only timing derate information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

The *report_timing_derate* command reports derating previously set by the *set_timing_derate* command for each object on which derating has been set: the design, cell instances, and nets; or for the objects listed in the *object_list* option. The derating factors are reported in a table format, organized by setting type:

```
pt_shell> report_timing_derate -significant_digits 2
 ...
                           ----- Clock -----              ----- Data -----
                          Rise          Fall            Rise
 Fall
                      Early  Late    Early  Late    Early  Late    Early
 Late
-------------------------------------------------------------------------
-------
design: MYDESIGN
  Net delay static    0.90   1.20    0.90   1.10    0.90   1.20    0.90
 1.10
  Net delay dynamic   0.90   1.20    0.90   1.10    0.90   1.20    0.90
 1.10
  Cell delay          0.90   1.20    0.90   1.10    0.90   1.20    0.90
 1.10
  Cell check           --     --      --     --      --     --      --
  --

cell (hier): DESIGN_TOP/I_RISC_CORE/I_ALU
  Net delay static    0.90   1.20    0.90   1.10    0.90   1.20    0.90
 1.10
```

```
  Net delay dynamic    0.90    1.20    0.90    1.10    0.90    1.20    0.90
 1.10
  Cell delay           0.85    1.20    0.85    1.10    0.85    1.20    0.85
 1.10
  Cell check            --      --      --      --      --      --      --
  --

net: net198
  Net delay static     0.90    1.30    0.90    1.30    0.90    1.30    0.90
 1.30
  Net delay dynamic    0.90    1.30    0.90    1.30    0.90    1.30    0.90
 1.30
```

To report the name of the design, ancestor cell, or library cell from which each object inherits its derating settings, use the *-include_inherited* option in the command. For example, a leaf-level cell might show the name of the ancestor cell or library cell on which derating was set.

Examples

In the following example, derating is set globally (only on the design), so the *report_timing_derate* command reports only global derating factors.

```
pt_shell> set_timing_derate  -data -early 0.92
pt_shell> set_timing_derate  -data  -late 1.14
pt_shell> set_timing_derate -clock -early 0.75
pt_shell> set_timing_derate -clock  -late 1.26
pt_shell> report_timing_derate
 ...
                              ----- Clock ------              ------ Data
 ------
                             Rise            Fall            Rise
   Fall
                         Early    Late   Early    Late   Early    Late
 Early    Late
 ------------------------------------------------------------------------
------------
 design: simple_path
   Net delay static      0.75    1.26    0.75    1.26    0.92    1.14
 0.92    1.14
   Net delay dynamic     0.75    1.26    0.75    1.26    0.92    1.14
 0.92    1.14
   Cell delay            0.75    1.26    0.75    1.26    0.92    1.14
 0.92    1.14
   Cell check             --      --      --      --      --      --
 --      --
```

In the following example, incremental derating is set globally (only on the design), so the *report_timing_derate -increment* command reports only global incremental derating adjustments.

Feedback

```
pt_shell> set_timing_derate  -data -increment -early -0.02
pt_shell> set_timing_derate  -data -increment  -late  0.06
pt_shell> set_timing_derate -clock -increment -early -0.05
pt_shell> set_timing_derate -clock -increment  -late  0.10
pt_shell> report_timing_derate -increment
 ...
                             ----- Clock ------              ------ Data
 ------
                           Rise            Fall            Rise
    Fall
                       Early    Late   Early    Late    Early    Late
 Early    Late
 ------------------------------------------------------------------------
-------------
 design: simple_path
   Net delay static    -0.05    0.10   -0.05    0.10    -0.02    0.06
 -0.02    0.06
   Net delay dynamic   -0.05    0.10   -0.05    0.10    -0.02    0.06
 -0.02    0.06
   Cell delay          -0.05    0.10   -0.05    0.10    -0.02    0.06
 -0.02    0.06
   Cell check            --      --      --      --       --      --
 --       --
```

In the following example, the commands set derating globally on the design TOP and more narrowly on the hierarchical block H1. The *report_timing_derate* command reports the derating factors that apply to the lower-level cell H1/U5.

```
pt_shell> set_timing_derate -early 0.95
pt_shell> set_timing_derate -late  1.06
pt_shell> set_timing_derate -early 0.82 [get_cells H1]
pt_shell> set_timing_derate -late  1.12 [get_cells H1]
pt_shell> report_timing_derate -include_inherited [get_cells H1/U5]
 ...

                             ----- Clock ------              ------ Data
 ------
                           Rise            Fall            Rise
  Fall
                       Early    Late   Early    Late    Early    Late
 Early    Late
 ------------------------------------------------------------------------
-----------
 cell (hier): H1/U5
   Net delay static    0.95    1.06    0.95    1.06    0.95    1.06
 0.95    1.06
   Inherited            TOP     TOP     TOP     TOP     TOP     TOP
 TOP     TOP

   Net delay dynamic   0.95    1.06    0.95    1.06    0.95    1.06
 0.95    1.06
```

```
  Inherited             TOP     TOP     TOP     TOP     TOP     TOP
TOP     TOP

  Cell delay           0.82    1.12    0.82    1.12    0.82    1.12
0.82    1.12
  Inherited             H1      H1      H1      H1      H1      H1
H1      H1

  Cell check            --      --      --      --      --      --
--      --
  Inherited             --      --      --      --      --      --
--      --
```

See Also

- set_timing_derate
- reset_timing_derate

---

report_timing_yield

This is a synonym for the *report_design_variation* command.

Syntax

int *report_design_variation*

```
design_variation_object
[-num_sigma value]
[-target_success_rate value]
[-nworst num_endpoints]
[-nosplit]
[-significant_digits digits]
```

See Also

- report_design_variation
- yield_enable_analysis
- get_timing_paths
- get_timing_yield
- report_yield_bottleneck
- report_cell_robustness
- report_voltage_robustness

Feedback

report_total_net_capacitance

Reports the total capacitance of specified nets.

Syntax

status *report_total_net_capacitance*

```
        nets
```

Data Types

nets	list

Arguments

nets

A list of one or more nets for which to report the total capacitance. At least one net is required. Glob-style wildcards (*) are supported.

Description

This command reports the total capacitance of specified nets.

This command is supported only for transistor-level GPDs.

Examples

The following example shows a total net capacitance report.

```
pt_shell> report_total_net_capacitance {SUM0 B0}
************************************
Report : Total Capacitance
Design : toprt
Version: Q-2019.12
Date   : Thu Nov 7 15:50:45 2019
************************************
========    =================
Net Name    Total Capacitance
========    =================
SUM0        0.013721
B0          0.089779
```

report_trace

Produce a report of performance profile metrics collected for traced commands.

Feedback

Syntax

report_trace [-start|-stop|resume] [-profile profile_type] [-command command_name] [-cumulative count] [-top count] [-quiet]

string *profile_type* string *command_name* int *count*

Arguments

`-start`

This option is mutually exclusive with -stop, -resume, -profile, -top, -cumulative, -command. The option starts the collection of performance profile statics for each traced command invocation. If metric collection had previously been stopped with -stop, then previously collected data are deleted before collection starts.

`-stop`

This option is mutually exclusive with -start,-resume, -profile, -top, -cumulative, -command. The option stops the collection of performance profile statics for traced command invocations.

`-resume`

This option is mutually exclusive with -start, -stop, -profile, -top, -cumulative, -command. The option resume the collection of performance profile statics for traced command invocations. Previous gathered statics remain and newly collected metrics are added to pre-existing data. This option may be used after -stop to resume data collection without deleting previously collected data.

`-profile` *`profile_type`*

This option is mutually exclusive with -start, -stop, and -resume. This option selects the profile metrics that are included in the report. Allowed valuesA valid value is a list of permitted values {memory, cpu, time, count}, or the value all to select all metrics. If -profile is given, then the report output contains performance profile data of given types only. If the option is omitted, then it defaults to all.

`-cumulative` *`count`*

This optional option is mutually exclusive with -start, -stop, -resume, and -command. This option limits the number of commands reported for the top resource consumer accumulated over all invocations. However, all commands that share the same usage metric as the minimum consumer included in the count will be shown. If the option is omitted, the report output contains the top 10 consumers.

`-top` *`count`*

This optional option is mutually exclusive with -start, -stop, -resume and -command. This option limits the number of commands reported for the top

resource consumer per single invocation. However, all commands that share the same usage metric as the minimum consumer included in the count will be shown. If the option is omitted, the report output contains the top 10 consumers.

`-command` *`command_name`*

This option is mutually exclusive with -start, -stop, -resume, -top and -cumulative. This option causes a performance profile report to be displayed for the given command. Cumulative and top single invocation metrics are shown. If the command was not invoked, then a message with this information is displayed instead of a report.

`-quiet`

If given, this option causes the command to ignore error conditions such as an attempt to start profile gathering when it has already been started or reporting on a non-existent command. The command will exit quietly when it encounters an error condition.

Description

The *report_trace* command collects performance profile metrics on all traced command invocations since the last -start. Stopping the gathering is not necessary to produce a report. When performance profile collection is started with report_trace -start, each traced command will be measured for the available performance metrics: memory, CPU, wallclock time, and call count. The accumulated profile data can be reported at any time by calling this command

Stopping the metric gathering with -stop freezes the gathered metrics until the next -start. When gathering is restarted, all previously gathered metrics are deleted. Alternatively, metric collection may be restarted without deleting previously collected data using -resume.

The *report_trace* with no options will report the top 10 consumers in both cumulative and single call metrics for all profile types collected thus far.

Examples

The following example starts the collection of performance profile metrics.

```
prompt> report_trace -start
on
```

The following example produces a report of top 10 commands with most usage for all profile types for both cumulative and individual call consumption. 10 is the default number reported for -top and -cumulative when these options are omitted.

```
prompt> report_trace
on
```

The following example produces a report of cumulative and the single most expensive invocation for each profile type for the command update_timing.

```
prompt> report_trace -command update_timing
on
```

The following example produces a report of top 20 consumers for cumulative and single invocation collected thus far.

```
prompt> report_trace -top 20 -cumulative 20
on
```

The following example produces a report of top 5 cumulative consumption of memory.

```
prompt> report_trace -profile memory -cumulative 5
on
```

The following example produces a report of top 15 commands with most invocations.

```
prompt> report_trace -profile count -cumulative 15
on
```

The following example produces a report of top 10 commands with most cpu and wallclock time usage for both cumulative and individual call consumption. 10 is the default number reported for -top and -cumulative when these options are omitted.

```
prompt> report_trace -profile {cpu time}
on
```

See Also

- set_trace_option
- get_trace_option
- log_trace
- annotate_trace

---

report_transitive_fanin

Reports logic in fanin of specified objects.

Syntax

string *report_transitive_fanin*

```
[-nosplit]
[-from from_list]
[-through through_list]
```

```
-to to_list
[-trace_arcs timing | enabled | all]
```

Data Types

```
from_list          list
through_list       list
to_list            list
```

Arguments

`-nosplit`

Does not split lines if columns overflow.

`-from from_list`

Specifies a list of pins, ports, and nets in the design to constrain the fanin of *to_list* that is reported. If you specify a net, the effect is same as listing all the load pins on the net.

`-through through_list`

Specifies a list of pins, ports, and nets in the design to constrain the fanin of *to_list* that is reported. If you specify a net, the effect is same as listing all the pins on the net.

`-to to_list`

Specifies a list of pins, ports, or nets in the design, whose transitive fanin is reported. If you specify a net, the effect is the same as listing all driver pins on the net.

`-trace_arcs timing | enabled | all`

Specifies the type of combinational arcs to trace during the traversal. Allowed values are

- *timing* (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- *enabled* - Permits the tracing of all enabled arcs and disregards case analysis values.
- *all* - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

Description

The command produces a report showing the transitive fanin of specified pins, ports, or nets in the design. A pin is considered to be in the transitive fanin of an object if there is a timing path through combinational logic from the pin to that object (also see the *-trace_arcs* option). The fanin report stops at the clock pins of registers (sequential cells).

If the *current_instance* command is set, the report focuses on the fanin within the *current_instance* command. The report stops at the boundaries of the *current_instance* command, and no paths outside the *current_instance* command are reported. The report shows the hierarchical boundary pins on the current instance.

Examples

The following example shows the transitive fanin of a pin in the design.

```
pt_shell> report_transitive_fanin -to FF1/D

****************************************
Report : transitive_fanin
Design : top
 ...
****************************************

Fanin network of sink 'FF1/D':

Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
gate1/Y          FF1/D             (net arc)   same


Load Pin         Driver Pin        Type        Sense
-------------------------------------------------------------
gate1/A          gate1/Y           AN2         same
gate1/B          gate1/Y           AN2         same


Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
IN1              gate1/A           (net arc)   same
IN2              gate1/B           (net arc)   same
```

See Also

- current_design
- current_instance
- report_clock
- report_timing
- report_transitive_fanout

---

report_transitive_fanout

Reports logic in fanout of specified objects.

Feedback

Syntax

string *report_transitive_fanout*

```
[-nosplit]
[-trace_arcs timing | enabled | all]
-clock_tree
-from from_list
[-through through_list]
[-to to_list]
```

Data Types

```
from_list          list
through_list         list
to_list            list
```

Arguments

`-nosplit`

Does not split lines if columns overflow.

`-trace_arcs timing | enabled | all`

Specifies the type of combinational arcs to trace during the traversal. Allowed values are

- *timing* (the default) - Permits tracing of valid timing arcs only -- that is, arcs that are neither disabled nor invalid due to case analysis.
- *enabled* - Permits the tracing of all enabled arcs and disregards case analysis values.
- *all* - Permits the tracing of all combinational arcs regardless of case analysis or arc disabling.

`-clock_tree`

Reports transitive fanout of all clock sources in the design. When this option is used, the fanout search is constrained to objects in the clock network only.

`-from from_list`

Specifies a list of pins, ports, and nets in the design whose transitive fanout is reported. If you specify a net, the effect is same as listing all the load pins on the net.

`-through through_list`

Specifies a list of pins, ports, and nets in the design to constrain the fanout of *from_list* that is reported. If you specify a net, the effect is same as listing all the pins on the net.

`-to` *to_list*

Specifies a list of pins, ports, and nets in the design to constrain the fanout of *from_list* that is reported. If you specify a net, the effect is same as listing all the driver pins on the net.

Description

Produces a report showing the transitive fanout of specified pins or ports in the design. A pin is considered to be in the transitive fanout of an object if there is a timing path through combinational logic from the object to that pin (also see the *-trace_arcs* option). The fanout report stops at the inputs to registers (sequential cells).

If the *current_instance* command is set, the report focuses on the fanout within the *current_instance* command. The report stops at the boundaries of the *current_instance* command, and no paths outside the *current_instance* command are reported. The report also shows the hierarchical boundary pins on the *current_instance* command.

Examples

The following example shows the transitive fanout of a pin in the design.

```
pt_shell> report_transitive_fanout -from FF1/Q

****************************************
Report : transitive_fanout
Design : top
 ...
****************************************

Fanout network of source 'FF1/Q':


Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
FF1/Q            gate5/A           (net arc)   same


Load Pin         Driver Pin        Type        Sense
-------------------------------------------------------------
gate5/A          gate5/Y           AN2         same


Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
gate5/Y          OUT2              (net arc)   same
```

The following command shows the transitive fanout of all the clock sources in the design, constrained to the clock network.

```
pt_shell> report_transitive_fanout -clock_tree
```

Feedback

```
****************************************
Report : transitive_fanout
         -clock_tree
Design : top
 ...
****************************************

Fanout network of source 'CLK':

Driver Pin       Load Pin          Type        Sense
-------------------------------------------------------------
CLK              FF3/CLK           (net arc)   same
CLK              FF2/CLK           (net arc)   same
CLK              FF1/CLK           (net arc)   same
```

See Also

- create_clock
- current_design
- current_instance
- report_clock
- report_timing
- report_transitive_fanin

---

report_units

Reports the unit information.

Syntax

string *report_units*

```
[-nosplit]
```

Arguments

`-nosplit`

Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column. The *-nosplit* option prevents line-splitting and facilitates writing software to extract information from the report output.

Description

The *report_units* command displays the units used by the current design. To generate the report, the design should be loaded and linked to a library. The units are always reported

Feedback

in reference to the MKS units such as Farad, Amp, Ohm, Second, and Volt. The resistance unit is inferred from the time and capacitance units. Both time and capacitance units are required for PrimeTime. If either of them is missing, the units are undefined and is reported as "Not specified". The most common cause of this problem occurs when the library is in an old format that needs to be recompiled with the latest Library Compiler.

If power analysis is on, the power units is also reported.

Examples

The following report is an example of the unit information that is reported:

```
pt_shell> report_units

****************************************
Report : units
Design : simple
Version: D-2010.06
Date   : Mon Jun 28 11:55:27 2010
****************************************

Units
------------------------------------------------
Capacitive_load_unit        : 1e-12 Farad
Current_unit                : 0.001 Amp
Resistance_unit             : 1000 Ohm
Time_unit                   : 1e-09 Second
Voltage_unit                : 1 Volt
```

See Also

- read_db
- set_min_library

---

report_user_def_clock_savings

Generates power saving reports by controlling the enable parameters (toggle rate, static probability) of the registers

Syntax

string *report_user_def_clock_savings*

```
[-clock_gate_type      clock_gate_value_type]
[-enable_def_type      enable_values_type]
[-enable_toggle_rate   toggle_rate_value]
[-enable_sp            static_probability_value]
[-min_pwr_savings      power_savings]
[-min_reg_width        register_width]
```

```
[-report_nl_names]
[object_list]
```

Data Types

clock_gate_value_type	string
enable_values_type	string
toggle_rate_value	float
static_probability_value	float
power_savings	float
register_width	integer
object_list	list

Arguments

`-clock_gate_type` *clock_gate_value_type*

Use this option to specify the type of clock which drives the registers. The clock could be gated type, domain type (non-gated) or both. Default value of *-clock_gate_type* is "all" and other supported values are "gated" and "not_gated".

`-enable_def_type` *enable_values_type*

Use this option to specify the type of activity data to be used. There could be two types of activity data "avg_defs" and "user_defs". In "avg_defs" activity data is the average of design and in "user_defs" activity data is explicitily defined by the user. Default value of *-enable_def_type* is "avg_defs".

`-enable_toggle_rate` *toggle_rate_value*

Use this option to specify the Toggle Rate for the enable Pin. This option is only applicable when *-enable_def_type* is "user_defs". Default value of *-enable_toggle_rate* is 2 and used only for the non-gated registers.

`-enable_sp` *static_probability_value*

Use this this option to specify the Staic Probability for the enable Pin. This option is only applicable when *-enable_def_type* is "user_defs". Default value of *-enable_sp* is 0.5 and used only for the non-gated registers.

`-min_pwr_savings` *power_savings*

Use this option to report only those registers whose total power saving, after using the power saving technique, is more than or equal to the "power_savings" value. Default value of *-min_pwr_savings* is 0.0.

`-min_reg_width` *register_width*

Use this option to filter the bundled registers so that the bundled registers having width greater than or equal to "register_width" will be considered for power saving estimate. Default value of *-min_reg_width* is 8.

`-report_nl_names`

Use this option to report the netlist(gate-level) name of the register. By default, the RTL name or merged RTL name, in case of bus register, is reported for the registers.

`object_list`

Use this option to apply the power savings techniques on the list of registers. Options *-min_reg_width* and *-min_pwr_savings* cannot be supported with this option. Currently only the gate-level register name can be specified.

Description

Use the *report_user_def_clock_savings* command to generate the power savings report after explicitly defining the activity data of Enable of register/bus in the design. Presently, the static probability and toggle rate parameters are defined by the user.

The two important fields that the command prints are the name of registers and the potential power savings that could happen using the above technique. The other fields are: width of register bus, frequency of clock, the register is gated or not, the current and final register gate efficiency percentage, current and final enable probability, current power and the final power.

To use the command, you should first apply the update_power and update_metrics command or apply the compute_metrics command.

Controlling Activity Data on Enable

For clock-gated register, it may happen that the enable signal is not optimal and hence its Q/CP ratio could be low. In order to find the optimal enable signal, the user would like to find its optimum activity data i.e. Toggle Rate and Static Probability. By trying different values of activity data of the enable, a user can find the optimum values where maximum power savings can be achieved for the given registers.

Similarly, for non-gated registers, a user may want to see the amount of power savings achieved by firstly making it clock-gated and then finding the optimum activity data for the enable in the same way as for gated-register.

Average and User-Defined Activity Data

In "avg_defs", the activity data is calculated from the average of activity data values of all the Enable signals existed in the design.

In "user_defs", the activity data values are explicitly defined by the user. The user can either define Toggle Rate or Static Probability or both. If the user defines only one value then the other value will be the: 1. Default value in case of non-gated register. 2. Original value of the signal in case of gated register.

Feedback

Register Name in report

The command prints the RTL register name or the merged RTL names in case of bus registers by default. If mapping information does not exist for some registers the gate level name would get printed instead.

To print the gate-level register names instead of the RTL names use the option -report_nl_names.

Examples

The following example shows a list of bundled registers having a width >=8 (default value) where the power could be saved by setting the average activity data to the enable of registers. The registers may have either a gated clock or not.

```
pt_shell> report_user_def_clock_savings -clock_gate_type all
 -enable_def_type avg_defs
****************************************
Report : report_user_def_clock_savings
Design : ethmac
Version: R-2020.09-SP5-2-VAL-20211101
Date   : Mon Nov  1 15:59:45 2021
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
Register                              Register            Gated
   Clock Pin           Current              Current             Current
           Final               Final               Final
 Predicted
Name                                  Width               (Y/N)
   Frequency           Register Gating     Enable              Power
           Register Gating     Enable              Power
 Clock Power

   (MHz)               Efficiency (%)      Probability         (W)
           Efficiency (%)      Probability         (W)
 Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
temp_wb_dat_o_reg[0:31]               32                  Not Gated
   100.0               0.0                 0.0                 4.771e-06
           65.3                0.4                 9.125e-07
 3.859e-06
wishbone/TxData_wb[0:31]              32                  Not Gated
   100.0               0.0                 0.0                 3.328e-06
           65.3                0.4                 9.124e-07
 2.415e-06
```

```
wishbone/rx_fifo/data_out_reg[0:31]
                                    32                  Not Gated
   100.0                0.0                  0.0                  3.328e-06
            65.3                0.4                  9.124e-07
 2.415e-06
txethmac1/random1/x_reg[0:9]        10                  Not Gated
   100.0                0.0                  0.0                  4.89e-07
             65.7                0.4                  2.95e-07
 1.941e-07
wishbone/ram_addr_reg[7:0]          8                   Gated
   100.0               49.0                  0.5                  7.734e-07
            65.3                0.4                  5.941e-07
 1.793e-07
txethmac1/txcrc/Crc[0:7, 16:31]     24                  Not Gated
   100.0                0.0                  0.0                  4.926e-07
            65.7                0.4                  3.374e-07
 1.552e-07
rxethmac1/crcrx/Crc[0:7, 16:23]     16                  Not Gated
   100.0                0.0                  0.0                  3.223e-07
            65.7                0.4                  1.843e-07
 1.379e-07
rxethmac1/Crc[8:15, 24:31]          16                  Not Gated
   100.0                0.0                  0.0                  3.223e-07
            65.7                0.4                  1.844e-07
 1.379e-07
txethmac1/Crc[8:15]                 8                   Not Gated
   100.0                0.0                  0.0                  4.009e-07
            65.7                0.4                  2.663e-07
 1.346e-07
rxethmac1/LatchedByte[0:7]          8                   Not Gated
   100.0                0.0                  0.0                  2.809e-07
            65.7                0.4                  1.47e-07
 1.339e-07
RxData[0, 2:6]                      8                   Not Gated
   100.0                0.0                  0.0                  2.666e-07
            65.7                0.4                  1.458e-07
 1.207e-07
wishbone/ram_di_reg[0:31]           32                  Gated
   100.0               49.0                  0.5                  1.11e-06
             65.3                0.4                  1.012e-06
 9.785e-08
txethmac1/txcounters1/NibCnt_reg[0:15]
                                    16                  Gated
   100.0                6.5                  1.0                  5.542e-07
            65.7                0.4                  5.25e-07
 2.914e-08
rxethmac1/RxData_d_reg[7:0]         8                   Gated
   100.0                0.8                  1.0                  6.024e-08
            65.7                0.4                  4.523e-08
 1.501e-08
maccontrol1/transmitcontrol1/ControlData_reg[7:0]
                                    8                   Gated
   100.0               -0.1                  1.0                  2.973e-07
```

```
           65.7                0.4                 2.863e-07
 1.101e-08
------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
----------------
1
```

The following example shows a list of bundled registers having a width >=16 where the power could be saved by setting the average activity data to the enable of registers. The registers may have either a gated clock or not. The use of the option *report_nl_names* has resulted in printing the netlist names of bundled register (gate-level).

```
pt_shell> report_user_def_clock_savings -clock_gate_type all
 -enable_def_type avg_defs -min_reg_width 16 -report_nl_names
****************************************
Report : report_user_def_clock_savings
Design : ethmac
Version: R-2020.09-SP5-2-VAL-20211101
Date   : Mon Nov  1 15:59:59 2021
****************************************

------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
----------------
Register                            Register            Gated
   Clock Pin           Current              Current              Current
           Final               Final                Final
 Predicted
Name                                Width                (Y/N)
   Frequency           Register Gating     Enable               Power
           Register Gating     Enable               Power
 Clock Power

   (MHz)               Efficiency (%)      Probability          (W)
           Efficiency (%)      Probability          (W)
 Savings (W)
------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
----------------
temp_wb_dat_o_reg_reg[0:31]         32                   Not Gated
   100.0               0.0                  0.0                  4.771e-06
           65.3                0.4                  9.125e-07
 3.859e-06
wishbone/rx_fifo/data_out_reg[0:31]
                                    32                   Not Gated
   100.0               0.0                  0.0                  3.328e-06
           65.3                0.4                  9.124e-07
 2.415e-06
wishbone/tx_fifo/data_out_reg[0:31]
```

```
                                    32                  Not Gated
   100.0                0.0                    0.0                   3.328e-06
            65.3                 0.4                   9.124e-07
 2.415e-06
txethmac1/txcrc/Crc_reg[0:7, 16:31]
                                    24                  Not Gated
   100.0                0.0                    0.0                   4.926e-07
            65.7                 0.4                   3.374e-07
 1.552e-07
rxethmac1/crcrx/Crc_reg[0:7, 16:23]
                                    16                  Not Gated
   100.0                0.0                    0.0                   3.223e-07
            65.7                 0.4                   1.843e-07
 1.379e-07
rxethmac1/crcrx/Crc_reg[8:15, 24:31]
                                    16                  Not Gated
   100.0                0.0                    0.0                   3.223e-07
            65.7                 0.4                   1.844e-07
 1.379e-07
wishbone/ram_di_reg[0:31]           32                  Gated
   100.0                49.0                   0.5                   1.11e-06
             65.3                  0.4                    1.012e-06
 9.785e-08
txethmac1/txcounters1/NibCnt_reg[0:15]
                                    16                  Gated
   100.0                6.5                    1.0                   5.542e-07
            65.7                 0.4                   5.25e-07
 2.914e-08
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
1
```

The following example shows a list of bundled registers having a width >=0 where the power could be saved by defining the Toggle Rate as 0.1 and Static Probability as 0.3 to the enable of registers. In this case, the activity data is applied to "gated" registers only.

```
pt_shell> report_user_def_clock_savings -clock_gate_type gated
 -enable_def_type user_defs -enable_toggle_rate 0.1 -enable_sp 0.3
 -min_reg_width 0
****************************************
Report : report_user_def_clock_savings
Design : ethmac
Version: R-2020.09-SP5-2-DEV-20211101
Date   : Tue Nov  2 13:47:46 2021
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
```

```
Register                              Register            Gated
   Clock Pin            Current              Current               Current
            Final                Final                Final
 Predicted
Name                                  Width                (Y/N)
   Frequency            Register Gating     Enable               Power
            Register Gating     Enable               Power
 Clock Power

   (MHz)                Efficiency (%)      Probability          (W)
            Efficiency (%)      Probability          (W)
 Savings (W)
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
----------------
wishbone/ram_addr_reg[7:0]            8                    Gated
   100.0                49.0                 0.5                   7.734e-07
            70.5                 0.3                  5.642e-07
 2.092e-07
wishbone/ram_di_reg[0:31]             32                   Gated
   100.0                49.0                 0.5                   1.11e-06
             70.5                 0.3                  9.308e-07
 1.793e-07
wishbone/tx_burst_cnt_reg[0:2]        3                    Gated
   100.0                0.6                  1.0                   6.446e-07
            70.5                 0.3                  4.853e-07
 1.593e-07
wishbone/TxEn_reg_RxEn_reg            2                    Gated
   100.0                49.0                 0.5                   5.786e-07
            70.5                 0.3                  4.357e-07
 1.429e-07
miim1/clkgen/Counter_reg[4]_Counter_reg[1:2]
                                      6                    Gated
   100.0                50.1                 0.5                   1.032e-06
            70.5                 0.3                  9.35e-07
 9.74e-08
miim1/clkgen/Mdc_reg                  1                    Gated
   100.0                50.1                 0.5                   8.801e-07
            70.5                 0.3                  8.045e-07
 7.564e-08
wishbone/BDWrite_reg[3:0]             4                    Gated
   100.0                49.4                 0.5                   7.28e-07
             70.5                 0.3                  6.688e-07
 5.919e-08
wishbone/BDRead_reg                   1                    Gated
   100.0                49.4                 0.5                   6.164e-07
            70.5                 0.3                  6.043e-07
 1.212e-08
-------------------------------------------------------------------------
-------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
---------------
1
```

The following example shows a list of bundled registers having a width >=0 where the power could be saved by defining the Toggle Rate as 0.1 and Static Probability as 0.3 to the enable of registers. In this case, the activity data is applied to "non-gated" registers only.

```
pt_shell> report_user_def_clock_savings -clock_gate_type not_gated
 -enable_def_type user_defs -enable_toggle_rate 0.1 -enable_sp 0.3
 -min_reg_width 0
****************************************
Report : report_user_def_clock_savings
Design : ethmac
Version: R-2020.09-SP5-2-DEV-20211101
Date   : Tue Nov  2 13:47:48 2021
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------
Register                              Register            Gated
   Clock Pin           Current              Current              Current
           Final               Final               Final
 Predicted
Name                                  Width                (Y/N)
   Frequency           Register Gating     Enable               Power
           Register Gating     Enable               Power
 Clock Power

   (MHz)               Efficiency (%)      Probability          (W)
           Efficiency (%)      Probability          (W)
 Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------
temp_wb_dat_o_reg[0:31]               32                   Not Gated
   100.0               0.0                  0.0                  4.771e-06
           70.5                0.3                  8.552e-07
 3.916e-06
wishbone/TxData_wb[0:31]              32                   Not Gated
   100.0               0.0                  0.0                  3.328e-06
           70.5                0.3                  8.456e-07
 2.482e-06
wishbone/rx_fifo/data_out_reg[0:31]
                                      32                   Not Gated
   100.0               0.0                  0.0                  3.328e-06
           70.5                0.3                  8.456e-07
 2.482e-06
```

```
ethreg1/INT_SOURCEOut[0, 2:3, 5:6] 5                    Not Gated
   100.0                0.0                  0.0                  2.376e-06
            70.5                0.3                  3.728e-07
 2.003e-06
miim1/EndBusy                        1                    Not Gated
   100.0                0.0                  0.0                  2.173e-06
            70.5                0.3                  2.684e-07
 1.905e-06
miim1/WCtrlData_q3_reg               1                    Not Gated
   100.0                0.0                  0.0                  2.173e-06
            70.5                0.3                  2.684e-07
 1.905e-06
wishbone/BlockReadTxDataFromMemory_reg
                                     1                    Not Gated
   100.0                0.0                  0.0                  2.165e-06
            70.5                0.3                  2.66e-07
 1.899e-06
wishbone/BlockingIncrementTxPointer_reg
                                     1                    Not Gated
   100.0                0.0                  0.0                  2.165e-06
            70.5                0.3                  2.66e-07
 1.899e-06
wishbone/RxOverrun                   1                    Not Gated
   100.0                0.0                  0.0                  2.165e-06
            70.5                0.3                  2.66e-07
 1.899e-06
wishbone/ShiftEndedSync3_reg         1                    Not Gated
   100.0                0.0                  0.0                  2.165e-06
            70.5                0.3                  2.66e-07
 1.899e-06
wishbone/ShiftEnded_reg              1                    Not Gated
   100.0                0.0                  0.0                  2.165e-06
            70.5                0.3                  2.66e-07
 1.899e-06
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
1
```

The following example shows a list of bundled registers having a width >=0 where the power could be saved by defining the Toggle Rate as 0.15 and Static Probability as 0.5 to the enable of registers. In this case, the activity data is applied to "gated" registers only.

```
pt_shell> report_user_def_clock_savings -clock_gate_type gated
 -enable_def_type user_defs -enable_toggle_rate 0.15 -enable_sp 0.5
 -min_reg_width 0
****************************************
Report : report_user_def_clock_savings
Design : ethmac
Version: R-2020.09-SP5-2-DEV-20211101
Date   : Tue Nov  2 13:47:50 2021
```

```
****************************************

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
Register                              Register            Gated
   Clock Pin           Current              Current             Current
           Final                Final                Final
 Predicted
Name                                  Width                (Y/N)
   Frequency           Register Gating     Enable               Power
           Register Gating     Enable               Power
 Clock Power

   (MHz)               Efficiency (%)      Probability          (W)
           Efficiency (%)      Probability          (W)
 Savings (W)
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
wishbone/tx_burst_cnt_reg[0:2]     3                    Gated
   100.0               0.6                  1.0                  6.446e-07
           50.4                0.5                 6.047e-07
 3.99e-08
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------
1
```

See Also

- compute_metrics
- update_power
- update_metrics
- report_power
- report_rtl_metrics
- report_est_power_savings

report_variation_bottleneck

Reports a list of cells with large high-sigma failure rate in a *design_variation* object created by the *get_design_variation* command.

Feedback

Syntax

int *report_variation_bottleneck*

```
design_variation_object
[-max_violators num_cells]
[-nosplit]
[-significant_digits digits]
```

Data Types

```
design_variation_object          collection
num_cells              int
digits                 int
```

Arguments

design_variation_object

Specifies a *design_variation* object collection, either an embedded *get_design_variation* command in square braces or a variable previously set to the result of a *get_design_variation* command. The *report_variation_bottleneck* command reports the variation bottleneck cells for this *design_variation* object.

`-max_violators` *num_cells*

Specifies the maximum number of cells to be reported. The default is 1000.

`-nosplit`

Prevents line-splitting to facilitate parsing of information from the report output. Most of the design information is listed in fixed-width columns. Without this option, if the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`-significant_digits` *digits*

Specifies the number of digits after the decimal point displayed for values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which is 2 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

Description

The *report_variation_bottleneck* command reports a list of cells with high-sigma failure rate in a *design_variation* object created by the *get_design_variation* command. The list is sorted from largest to smallest high-sigma failure rate.

You can use this list for robustness ECO fixing. Replacing some of the top cells having the largest high-sigma failure rate with lower-Vt cells can help reduce design high-sigma failure rate significantly.

To perform variation bottleneck analysis, the *ps_enable_analysis* variable must be set to *true*.

Examples

The following example shows the typical usage of *report_variation_bottleneck* command.

```
prompt> set mypaths [get_timing_paths -pba_mode exhaustive \\
          -path_type full_clock_expanded -nworst 10 \\
          -max_paths 10000 -pocv_pruning -slack_lesser_than 1.0]

prompt> set mydvar [get_design_variation -sample_size 100000 $mypaths]

prompt> report_variation_bottleneck -max_violators 10 -nosplit $mydvar \\
          -significant_digits 6

   Cell         Lib_cell    Variation  Sigma_ratio  HSFR
   ------------------------------------------------------
   U10/U1897    BUFD1_S     1.678113   0.812944    0.000193
   U12/U12524   BUFD0_S     0.857391   0.548509    0.000111
   U18/U18778   AOID1_S     2.256163   0.830731    0.000077
   U20/U17180   AOID2_S     2.257476   0.831172    0.000077
   U16/U16837   AOID3_S     2.240766   0.827820    0.000076
   U11/U640     INVD1_S     1.488405   0.583223    0.000072
   U29/U708     AOID2_L     2.031563   0.796475    0.000069
   U35/U19558   AO2D1_S     2.251250   0.825218    0.000051
   U41/U4909    INVD0_S     1.492606   0.585177    0.000044
   U68/U1090    AOID3_S     1.610532   0.844050    0.000043
```

The variation bottleneck report has five columns:

- *Cell*: The cell instance with critical high-sigma failure rate.
- *Lib_cell*: The library cell name of the instance.
- *Variation*: The cell delay variation, which is the cell arc POCV corner delay minus the nominal delay.
- *Sigma_ratio*: The ratio of cell delay sigma to pin slack sigma.
- *HSFR*: The high-sigma failure rate of the cell instance.

The high-sigma failure rate is evaluated when a cell is at high-sigma while other cells have normal POCV variation. There may be many paths going through this cell. The reported high-sigma failure rate is the accumulated total considering all these paths. The cell list is sorted from largest to smallest high-sigma failure rate.

Often a small number of cells contribute to a large part of the total high-sigma failure rate. Therefore, performing timing ECOs on a few cells at the top of the list can improve overall joint success rate significantly.

Feedback

See Also

- ps_enable_analysis
- get_timing_paths
- get_design_variation
- report_design_variation
- report_cell_robustness
- report_voltage_robustness

---

report_vcd_hierarchy

Reports the hierarchy in the VCD file.

Syntax

```
status report_vcd_hierarchy
   [-pipe_exec shell_command_string]
   [-full_path]
   [-find pattern_string]
   [file_name]
```

Data Types

```
shell_command_string     string
pattern_string           string
file_name                string
```

Arguments

`-pipe_exec shell_command_string`

Specifies a shell command which is used to generate the VCD file *file_name*. This option invokes the command and directly pipe the output VCD file to PrimePower. In another word, the simulation and power analysis are in parallel run. No VCD disk file is generated at all.

`-full_path`

Displays each scopes in full path format.

`-find pattern_string`

Report only those scopes whose last hierarchy name is matched the specified *pattern*. They are always reported in full path format. Nothing is printed out if not found.

file_name

Specifies a file in VCD format from which the switching activity information is to be read. If file_name ends with .gz, .Z, .vpd and .fsdb, it is read as a gzipped file, a compressed file, a VCD+ file, and a FSDB file, respectively, otherwise it is just read as a normal ASCII VCD file. If the *file_name* is omitted, the name is taken from the *read_vcd* command if it exists; otherwise, an error message is issued.

Description

This command displays the indented hierarchy in the specified VCD file.

To read a gzip VCD file, a compressed VCD file, VCD+ file, or FSDB file, your PATH needs to include gunzip, uncompress, vpd2vcd, and fsdb2vcd. vpd2vcd and fsdb2vcd come with Synopsys VCS and Novas Debussy, respectively.

The pipe option allows PrimePower reading in VCD file directly from simulator, which avoids generating huge VCD file. The VCD file name in this scenario is just served as a pipe name. After running the VCD file does not exist. You don't need to change your existing test bench. Use it just like normal VCD dump (inserting $dumpfile, $dumpvars, etc.) You do not need to invoke the simulator, either. PrimePower automatically runs the simulation and directly read in the VCD output from the simulator.

Examples

The following example shows a VCD hierarchy report.

```
pt_shell> report_vcd_hierarchy vcd.dump.gz
tb
  macinst
    U3
    U4
    U7
    U9
    U10
    U11
    U12
```

See Also

- read_vcd

---

report_voltage_robustness

Generates a list of slack-critical or near-critical cells that have higher sensitivity to voltage rail (IR) drop.

Syntax

integer *report_voltage_robustness*

```
[-max_violators count]
[-nosplit]
[-pba]
[-dvd]
[-verbose]
[-reg_to_reg]
[-significant_digits digits]
[-sort_by slack | delay]
[-voltage_shift value]
[-voltage_shift_ratio value]
[-slack_lesser_than maximum_robustness_slack]
[-slack_margin margin_value]
[-mim_filter none | logical | timing]
[-cells cell_list]
[-plot]
[-prefix string]
```

Data Types

count	integer
digits	integer
value	float
maximum_robustness_slack	float
margin_value	float
cell_list	list

Arguments

`-max_violators` *count*

Specifies the maximum number of cells to be reported. The default is 10000.

`-nosplit`

Prevents line-splitting to facilitate the parsing of information from the report output. Most of the design information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

`-pba`

Performs voltage robustness using path-based analysis (PBA) timing, where on-demand exhaustive PBA pin slacks are used to avoid pessimism. The default is to use graph-based analysis (GBA) timing.

`-dvd`

Performs voltage robustness using the dynamic voltage drop (DVD) information read-in previously using the *read_dvd* command. The variable *timing_enable_dvd_analysis* must be set to *true* for the DVD voltage robustness analysis to be performed. An additional column “Delta_V” is reported in the voltage robustness report to show drop in voltage for each cell based on the

DvD file. Please note that "-dvd" cannot be used with "-voltage_shift" or "-voltage_shift_ratio" options.

`-verbose`

Shows detailed information on voltage robustness violators in the report.

`-reg_to_reg`

Restricts analysis to register-to-register paths.

`-significant_digits` *`digits`*

Specifies the number of digits after the decimal point displayed for values in the generated report. Allowed values are 0-13; the default is determined by the *report_default_significant_digits* variable, which is 2 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

`-sort_by slack | delay`

Specifies how the reported cells are sorted. The default is *slack*, which sorts by robustness slack. A value of *delay* sorts by delay impact instead.

`-voltage_shift` *`value`*

Specifies the voltage drop used for the sensitivity calculation, as an absolute value in volts. The default is a 5% relative drop from each cell's nominal VDD value.

`-voltage_shift_ratio` *`value`*

Specifies the voltage drop used for the sensitivity calculation, as a relative ratio (such as 0.95) of each cell's nominal VDD value. The default is a 5% relative drop.

`-slack_lesser_than` *`maximum_robustness_slack`*

Reports only cells with robustness slack less than the specified *maximum_robustness_slack* value; these cells have a negative robustness slack worse than the specified *maximum_robustness_slack* value (or a positive slack that is closer to causing a violation).

The default *maximum_robustness_slack* value for this option is 0.0.

`-slack_margin` *`margin_value`*

This option can be used to filter cells from robustness report. The cells whose worst timing slack (GBA or PBA), amongst paths passing through cell pins, is above the *margin_value* are filtered out from the report.

The default *margin_value* value for this option is infinity (that is, no filtering margin is applied).

`-mim_filter none | logical | timing`

Specifies how the reported cells are filtered out based on multiply intantiated modules (MIMs) grouping. This option can be used to increase the coverage in the report, given a *max_violator* count. The default value is *none*, that is, no MIM grouping is considered. When *logical* is specified, only the instance with worst robustness slack, amongst all logically shared MIM instances, is included in the report. With *timing*, the worst, amongst all MIM instances that share delays, will be reported.

`-cells` *`cell_list`*

Specifies a list of one or more cells in the current design to be analyzed. The analysis is only performed on the cells in the list.

`-plot`

Generates Gnuplot data and script file for the visualization of the robustness metrics.

`-prefix` *`string`*

Use with the *-plot* option. If a prefix string is provided using this option, this string is preprended to the generated Gnuplot file names.

Description

Cells with high voltage sensitivity can cause large timing differences for small changes in voltage, resulting in *Vmin*-critical paths. It is useful to identify these cells early in the design flow.

Voltage robustness analysis identifies the cells in the design that are sensitive to voltage drop. The *report_voltage_robustness* command performs this voltage robustness analysis.

You can specify the voltage shift as an absolute value or a relative drop from the nominal operating voltage. The default is a 5% relative drop from nominal VDD.

Voltage robustness analysis requires the following:

- PrimeShield variation analysis is enabled:

  ```
  prompt> set_app_var ps_enable_analysis true
  ```

- Pin slacks are saved during *update_timing*:

  ```
  prompt> ## set this before update_timing
  prompt> set_app_var ps_enable_save_timing_slack_data true
  ```

- True voltage scaling groups are defined with the *define_scaling_lib_group* command. For details and recommendations, see the *define_scaling_lib_group* command man page.

  The *report_voltage_robustness* command will issue a warning if the shifted voltage value exceeds the scaling range. Cells with exact-match scaling are skipped.

The resulting voltage robustness report shows a "Robustness Slack" value for each cell, which indicates the final slack with the voltage shift applied. The report is sorted based on this robustness slack by default.

Examples

The following example shows how to generate a voltage robustness report.

```
## Must save timing and arrival data
pt_shell> set ps_enable_analysis true
pt_shell> set ps_enable_save_timing_slack_data true

pt_shell> define_scaling_lib_group { ... }
pt_shell> set_voltage ...

pt_shell> update_timing -full

pt_shell> report_voltage_robustness -max_violators 1 \\
           -voltage_shift 0.05 -nosplit
```

The following output is a voltage robustness violation report for the previous example.

```
voltage_robustness
                                Timing     Delay      Robustness
   Cell               Lib_cell  slack      impact     slack
   ---------------------------------------------------------------
   uA/uB/U38            NOR2    -3.5523    0.0321     -0.0321
```

The robustness report shows the delay impact and robustness slack for each violator. The report is sorted based on the robustness slack by default.

The "Delay impact" column shows the impact on cell delay when voltage is shifted by the value specified.

```
Timing slack = slack at nominal VDD
Delay impact = (cell delay at shifted VDD) - (cell delay at nominal VDD)
```

The "Robustness slack" column shows the new slack with the adjusted cell delay.

```
Robustness slack = (slack at nominal VDD) - (Delay impact)
```

If the slack at nominal VDD is negative, then even cells with a small delay impact can be reported with negative robustness slack. This can mask voltage-drop bottleneck cells with high delay impact on marginally passing paths.

To avoid missing any true voltage-drop bottleneck cells in these cases, the tool uses *0* as the slack for negative-slack paths at nominal VDD, for the robustness slack calculation.

```
Robustness slack = 0 - (Delay impact)
```

The preceding report has a negative slack of -3.552 ns. In this case, with delay impact of 0.0321 ns, the robustness slack is calculated as -0.0321 ns.

See Also

- define_scaling_lib_group
- get_timing_paths
- ps_enable_analysis
- ps_enable_save_timing_slack_data
- timing_save_pin_arrival_and_required
- timing_save_pin_arrival_and_slack
- timing_enable_dvd_analysis
- read_dvd
- remove_dvd

---

report_waveform_integrity_analysis

Reports static or dynamic constraints for waveform integrity analysis.

Syntax

int *report_waveform_integrity_analysis*

```
[-static]
[-dynamic]
[-nosplit]
[-significant_digits num_digits]
[-sms_scenarios sms_scenarios_list]
[object_list]
```

Data Types

num_digits	integer
object_list	list
sms_scenarios_list	collection

Feedback

Arguments

`-static`

Reports the static constraint.

`-dynamic`

Reports the dynamic constraint.

`-nosplit`

Prevents line splitting in the report when columns overflow.

`-significant_digits` *`num_digits`*

Specifies the number of digits displayed in the report.

`object_list`

Specifies a list of cell input pins.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios to analyze. Only waveform integrity information compatible to the specified SMS scenarios collection is reported. This option is only available with SMVA or SMC analysis. This collection is created by *get_sms_scenarios*.

Description

This command reports the constraint limit for waveform integrity analysis set by the *set_waveform_integrity_analysis* command.

If the list of cell input pins are provided, the constraint limit on the specified cell inputs is reported. If there is no list of cell input pins, all constraints of both design-level and pin-level are reported.

Examples

The following example shows static constraints for waveform integrity analysis.

```
pt_shell> report_waveform_integrity_analysis -static
  ...

   Cell pins                                                  Tolerance
 ------------------------------------------------------------------------
----
   DFF1/CLK                                                     0.3000
   DFF1/D                                                       0.3000
   DFF2/CLK                                                     0.3000
   DFF2/D                                                       0.3000
```

Feedback

See Also

- remove_waveform_integrity_analysis
- report_constraint
- set_waveform_integrity_analysis

---

report_wire_load

Reports wire load information.

Syntax

string *report_wire_load*

```
[-nosplit]
[cell_list]
```

Data Types

```
cell_list     list
```

Arguments

`-nosplit`

Does not split lines if the column overflows.

cell_list

Specifies a list of hierarchical cells.

Description

The *report_wire_load* command shows the characteristics of wire load models set on the current design and on cells of the current design. If you do not specify a cell list, by default the command displays the wire load models and wire load mode used in the current design. If you specify a cell list, the command displays the wire load models used in the specified cells.

Examples

This command displays the wire load model in the current design.

```
pt_shell> report_wire_load

****************************************
Report : wire_load
Design : top
 ...
****************************************
```

Feedback

```
Wire Loading Model Mode: segmented

Cell           Design       Wire_model  Library       Selection_type
-------------------------------------------------------------------
               inv_top      WLM1        my_lib        user-specified
cel1           tst          WLM2        my_lib        user-specified

1
```

See Also

- remove_wire_load_model
- remove_wire_load_selection_group
- report_design
- set_wire_load_min_block_size
- set_wire_load_model
- set_wire_load_selection_group

---

report_yield_bottleneck

This is a synonym for the *report_variation_bottleneck* command.

Syntax

int *report_variation_bottleneck*

```
design_variation_object
[-max_violators num_cells]
[-nosplit]
[-significant_digits digits]
```

See Also

- report_variation_bottleneck
- yield_enable_analysis
- get_timing_paths
- get_timing_yield
- report_timing_yield
- report_cell_robustness
- report_voltage_robustness

Feedback

reset_activity_derate

Resets the clock activity derating factors for a cell or a specified list of instances.

Syntax

```
int reset_activity_derate
[-cell_list]
```

Data Types

cell_list	list

Arguments

`-cell_list`

the *derate_value* specified should be reset to cell instances specified in the object_list.

Description

Resets toggle rate derating factors currently set on a cell instance or the list of cells in the current design.

Clock activity derating factor affects toggle rate on cells in the clock network. The clock derating factors are used as a scalar multiplicative factor in calculation of toggle rate for output net of cell. If clock derating factors are not specified, the value of 1.0 is assumed.

The *cell_list* option can be used to reset derating factors on specific instances (cells) in the design.

After a reset, the derating factor goes back to the default value (derating factor of 1.0 for every instance in the design).

Examples

The following example resets clock activity toggle rate derating factor on all cells in the design.

```
pwr_shell> reset_activity_derate [get_cells *]
reset_activity_derate [get_cells *]
1
```

The following example resets the derating factor on cell instance "i_and_1" in the design.

```
pwr_shell> reset_activity_derate [get_cell i_and_1]
reset_activity_derate [get_cell i_and_1]
1
```

Feedback

See Also

- update_power
- set_switching_activity
- report_switching_activity

---

reset_aocvm_table_group

The *reset_aocvm_table_group* command removes the association between a named AOCV table set and a hierarchical instance or between a library cell and a AOCV derating group.

Syntax

status *reset_aocvm_table_group*

`[object_list]`

Data Types

object_list list

Arguments

object_list

Specifies the list of design objects.

Description

The *reset_aocvm_table_group* command removes the association between a named AOCV table set and a hierarchical instance or between a library cell and a AOCV derating group, where such an association was previously specified with the *set_aocvm_table_group* command.

Examples

The following example specifies an association between the hierarchical cell H1 and the AOCV table set A1 and then removes this association. After the association is removed, cells (or nets) fully enclosed within H1 derive their AOCV derating from a higher level hierarchical cell that fully encloses H1 and has a named AOCV table set associated with it. If no such hierarchical cell is found, the unnamed AOCV table set is used.

```
pt_shell> set_aocvm_table_group A1 [get_cells H1]

pt_shell> reset_aocvm_table_group [get_cells H1]
```

Feedback

See Also

- get_timing_paths
- read_aocvm
- remove_aocvm
- report_aocvm
- report_timing
- set_aocvm_coefficient
- set_aocvm_table_group
- write_binary_aocvm

---

reset_cell_mode

Resets cell mode groups to the default state.

Syntax

```
status reset_cell_mode
   [instance_list]
```

Data Types

instance_list	list

Arguments

instance_list

Specifies a list of instances for which the specified cell mode groups are to be set to default. If no instance list is specified, all moded cells are considered. This list must only be included with the *-type cell* option. No error occurs if you reset the modes on a cell that has no modes.

Description

This command resets mode groups to the default state of all modes enabled. Cell mode groups are defined in the library. Each library cell can have a set of cell mode groups. Each of these cell mode groups can have two or more cell modes. Each of these cell modes can be mapped to a set of timing arcs of the library cell.

When a cell mode group is set to default for a given instance of the library cell all of its modes are enabled for that cell. All timing arcs of the enabled modes also get enabled with respect to modes for that cell.

Feedback

Cell mode groups can have different states due to the *set_cell_mode*, *set_mode -type design*, and conditional evaluation. When conflict arises the following precedence rule is employed:

- *set_cell_mode*
- *set_mode -type design*
- Evaluation of mode conditions

To reset the state of the cell mode group due to the *set_cell_mode* command, use the *reset_cell_mode* command.

To reset the state of the cell mode group due to the *set_mode -type design* command, use either the *reset_mode -type design* or *remove_design_mode* command.

To reset the state of the cell mode group due to conditional evaluation, use the *remove_case_analysis* command or edit the Verilog netlist to remove logical constants.

You define design modes and design mode groups using the *define_design_group* and *map_design_mode* commands. Design modes can be mapped to a set of cell modes, set of paths, or both cell modes and paths. When a design mode group is reset to default all design modes of that group are enabled. When a design mode is enabled all paths mapped to the design mode are reset and all previous setting of cell modes mapped to the design modes are removed.

Examples

To reset all cell mode group in Uram1/DO, use this command.

```
pt_shell> reset_cell_mode Uram1/D0
```

To reset all cell mode groups in the design, use this command.

```
pt_shell> reset_cell_mode
```

See Also

- define_design_mode_group
- remove_design_mode
- report_mode
- report_cell_mode
- map_design_mode
- set_mode

Feedback

- set_cell_mode
- set_case_analysis

---

reset_clock_gate_max_power

Reset the vectorless mode for clock gate and memory cells, and reset flag for each of instances

Syntax

```
int reset_clock_gate_max_power
[-instance_list]
```

Data Types

instance_list	list

Description

Find the worst-case internal energy when-condition in the liberty model and set those for each of instances Ignore all propagated TR and SP values to the nets which are part of the when-condition If there is logic conflict with setting it will be ignored. For all other pins which are not part of the selected when-condition use the propagated TR and SP.

- PrimePower will use user defined conditions for power calculation for specific instances • Power calculation – Maximum of weighted sum of states that match the condition specified by user. – No change to toggle rate propagation • User constraint form (UI preliminary subject to change) – Set inst_power_condition <inst_name> {Pin_name bool_value derate_value,…} <related_pin> – inst_name and related_pin is mandatory – Ex: Set inst_power_condition top/blockA/mem1 {RCS 1 0.5, WCS 1 0.5} CLK – User does not need to specify conditions on all the other pins that are not included in the inst_power_condition command – Sum of derate values in an inst_power_condition cannot exceed 1.0 – If multiple inst_power_conditions defined for an instance, then tool will pick last statement one for power analysis for the same inst_name and related_pin

Example1: Single port memory – reset_clock_gate_max_power top/A/InstSp1 {“CS & !WE & !PD & SLEEPB” 1 0.5, “CS & WE & !PD & SLEEPB” 1 0.0} CK – Matching states – S2 = “CS & !WE & !PD & SLEEPB” 1 0.5 with CK S2_derate = 0.5 – S1 = “CS & WE & !PD & SLEEPB” 1 0.0 with CK S1_derate = 0.0 – total_derate_matching_states = 0.5 + 0.0 = 0.5 – Non_matching states – S3 = !CS & !PD & SLEEPB – total_derate_non_matching_states = 1.0 – total_derate_matching_states = 1-0.5=0.5 – Power calculation for supply VCC – matching_state_pwr – S2_pwr*S2_derate + S1_pwr*S1_derate = 40.0*0.5 + 50.0*0.0 = 20 – Non_matching_state_pwr – S3_pwr * total_derate_non_matching_states = 3.0*0.5 = 1.5 – Total_pwr_vcc = matching_state_pwr + non_matching_state_pwr = 20 + 1.5 = 21.5 – Similar calculation will be performed for cvcc – total_power = total_pwr_vcc + total_power_cvcc

Feedback

This command should only be run in average mode. update_power should be used to calculate power and report_power shows the maximum power values used for memory cells.

The *state_condition* option can be used to change the when condition of the state used for derating.

The *target_ts* option can be used to set the derate value to be used for a given state condition.

The *pin* option can be used to specify the input pin which needs to be used for derating and power calculation.

Examples

The following example shows how to set maximum memory power on a single cell in the design. The command dumps out the output in the report_power command.

```
pwr_shell> reset_clock_gate_max_power I_SINGLE_PORT -target_ts 0.5
reset_clock_gate_max_power I_SINGLE_PORT -target_ts 0.5

1
```

The following example sets the derate value of 0.5 for cell "I_TCAM" and creates an output file "debug_s1_report.rpt"

```
pwr_shell> reset_clock_gate_max_power I_TCAM -target_ts 0.5

reset_clock_gate_max_power I_TCAM -target_ts 0.5
1
```

See Also

- update_power
- report_power

---

reset_ctpm

Read CTPM file into current design for PrimeShield SPICE2Design feature.

Syntax

string *reset_ctpm*

```
[-library baselib_list]
```

Data Types

```
baselib_list            collection
```

Feedback

Arguments

`-library` *`baselib_list`*

Specifies the library list to reset ML-CTPM. Without the option, all CTPMs in design will be reset.

Description

The command *reset_ctpm* is used in PrimeShield SPICE2Design feature to reset any CTPM in design. This feature enables analysis for timing and power impact assessment of a given target. This target can be defined through a spice model representing next PDK, spot model, different spice corner, etc.

Through ML, Project Sicily aims to bridge the gap between the base and the target by capturing and gap through spice process parameters placed inside of a Compact Timing Power Model (CTPM) database. When CTPM is consumed in a base STA session, the QoR of this session is shifted such that to match QoR of the target.

After generating the CTPMs, you can use them in a real design to improve the QoR, by reducing the gap between the base library and the target PDK the CTPM generated from. The *read_ctpm* command reads CTPM generated by *gen_ctpm* into design. If base library QoR is needed after *read_ctpm*, use *reset_ctpm* to reset.

To perform a static timing analysis using a generated CTPM, perform the following steps: 1) Enable the PrimeShield tool for the CTPM generation using the following settings: *set ps_enable_analysis true set ps_enable_spice2design_analysis true* 2) Define the mapping between the base library already loaded in the design and its augmented library. See the following example: *define_sensitivity_lib_mapping [get_lib lib1] -side_lib lib1_side_file.db* 3) Read CTPM generated based on target PDK into the design. See the following example: *read_ctpm lib1.ctpm* 4) Peform complete STA: *update_timing -full*

Examples

The following script uses CTPM generated from target SPICE PDK to improve STA QoR comparing to target PDK, then reset CTPM to revert to original QoR:

```
set link_path orig.db
read_verilog ...
link_design
read_parastics ...
read_sdc ...
update_timing -full
# original STA results corresponding to original SPICE PDK

set ps_enable_analysis true
set ps_enable_spice2design_analysis true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]
read_ctpm lib1.ctpm
update_timing -full
# STA results shifted to target SPICE PDK
```

```
reset_ctpm
update_timing -full
# original STA results corresponding to original SPICE PDK
```

See Also

- gen_ctpm
- read_ctpm
- report_ctpm
- ps_enable_spice2design_analysis

---

reset_design

Removes user specified information from design.

Syntax

string *reset_design*

```
[-keep_parasitics]
```

Arguments

`-keep_parasitics`

Keeps the parasitics information on the current design.

Description

Removes all attributes from the design. It does not matter whether these attributes are user-defined, command implemented, or read from the design database. Attributes include, but are not limited to, exceptions, input/output delays, and clocks.

Examples

The following command resets the attributes on the current design.

```
pt_shell> reset_design
```

See Also

- current_design
- remove_user_attribute

Feedback

reset_eco_options

Resets options specified by the *set_eco_options* command.

Syntax

status *reset_eco_options*

Arguments

None.

Description

The *reset_eco_options* command removes all the ECO options specified by the *set_eco_options* command.

Distributed Multi-Scenario Analysis

To execute the *reset_eco_options* command in all scenarios, use the *remote_execute* command. For example:

```
remote_execute -verbose {
  reset_eco_options
}
```

Note: You cannot execute the *reset_eco_options* command in the manager process.

Examples

The following example shows how to use the *reset_eco_options* command.

```
pt_shell> reset_eco_options
```

See Also

- report_eco_options
- set_eco_options

reset_eco_scenarios

Reset any forced or ignored violation types set by the *set_eco_scenarios* command from the specified scenarios.

Syntax

```
status reset_eco_scenarios
scenario_name_list
```

Feedback

Data Types

scenario_name_list list

Arguments

scenario_name_list

Specifies one or more scenarios to which the specifications are removed.

Description

The *reset_eco_scenarios* removes any forced or ignored violation types set by the *set_eco_scenarios* command from the specified scenarios.

Note that this command affects only the live and static scenario analysis performed during the *start_eco_scenarios* or *report_eco_scenarios* command; it does not affect what data is used by the *fix_eco_** commands themselves.

Examples

The following example considers only setup timing violations in ss1, ss2, ss3 and ss4 scenarios, and then removes the specification from ss1 and ss2 scenarios.

```
pt_shell> set_eco_scenarios -only -type {setup} {ss1 ss2 ss3 ss4}
pt_shell> reset_eco_scenarios {ss1 ss2}
```

See Also

- set_eco_scenarios
- report_eco_scenarios
- start_eco_scenarios

---

reset_eco_wire

Resets ECO wire optimization changes back to the pre-ECO values.

Syntax

```
integer reset_eco_wire
[-all]
[net_list]
```

Data Types

net_list list

Feedback

Arguments

`-all`

Resets all ECO wire optimization changes back to pre-ECO values.

`net_list`

Lists of nets to reset the ECO wire optimization changes.

Description

This command resets ECO wire optimization changes performed by *fix_eco_wire* or *set_routing_rule* commands.

Examples

The following example resets the ECO wire optimization change to a net named *n43* back to the pre-ECO values.

```
prompt> reset_eco_wire {n43}
```

See Also

- fix_eco_wire
- report_eco_wire

---

reset_emmp_configuration

Specifies the reset configuration options for a hierarchical cell.

Syntax

status *reset_emmp_configuration*

`hier_instance_name`

Data Types

`hier_instance_name`	`string`

Arguments

`hier_instance_name`

To specify the hierarchical cell for which reset emmp config is to be done.

Feedback

Description

The *reset_emmp_configuration* command specifies resets logical memory config for the hierarchy. Which is set previously using *set_emmp_configuration* commands.

Examples

The following examples shows reset memory configuration. Here mem is the hierarchy cell.

```
pt_shell> reset_emmp_configuration mem
```

See Also

- set_emmp_configuration
- estimate_memory_max_power

---

reset_extract_model_indexes

Resets indexes on specified ports for extracted timing model (ETM) generation.

Syntax

```
status reset_extract_model_indexes
```

Arguments

None.

Description

This command replaces all user-specified index values on all ports with automatically-computed index values. The index values are used during model extraction.

See Also

- extract_model
- set_extract_model_indexes

---

reset_gpd_config

Cancels custom GPD configuration options previously set by the *set_gpd_config* command.

Syntax

string *reset_gpd_config*

`-gpd gpd_directory`

Data Types

`gpd_directory` `string`

Arguments

`-gpd gpd_directory`

Specifies that path to the GPD directory.

Description

The *reset_gpd_config* command removes custom GPD configuration options previously set by the *set_gpd_config* command and restores the GPD configuration options to their original states, as defined in the original GPD database generated by the StarRC tool.

Examples

The following session sets a coupling capacitor filtering option for the GPD database, reports the setting, and then resets the configuration back to the original.

```
set_gpd_config -gpd my_design1.gpd -relative_coupling_threshold 0.05
report_gpd_config -gpd my_design1.gpd
 ...
reset_gpd_config -gpd my_design1.gpd
report_gpd_config -gpd my_design1.gpd
```

See Also

- report_gpd_properties
- report_gpd_config
- set_gpd_config

---

reset_hier_resource_limits

Resets constraints specfied by *set_hier_resource_limits* command.

Syntax

status *reset_hier_resource_limits*

Arguments

None.

Feedback

Description

The *reset_hier_resource_limits* command removes all the constraints specified by the *set_hier_resource_limits* command.

```
This should be invoked prior to invocation of report_hier_analysis to
 report
load balanced partitions
```

Examples

The following example shows how to use the *reset_hier_resource_limits* command.

```
pt_shell> reset_hier_resource_limits
pt_shell> report_hier_analysis -explore_partitions
```

See Also

- set_hier_resource_limits
- report_hier_analysis

---

reset_implement_options

Resets options specified by the *set_implement_options* command.

Syntax

```
status reset_implement_options
   [-force]
   [-icc2_pre_legalize_script]
   [-icc2_eco_legalize_script]
   [-icc2_pre_route_script]
   [-icc2_eco_route_script]
   [-icc2_pre_extract_script]
   [-insert_fillers]
```

Arguments

`-force`

Forces the reset, discarding any unsaved pending design changes.

`-icc2_pre_legalize_script`

Removes the option specified by the *-icc2_pre_legalize_script* option of the *set_implement_options* command.

`-icc2_eco_legalize_script`

Removes the option specified by the *-icc2_eco_legalize_script* option of the *set_implement_options* command.

`-icc2_pre_route_script`

Removes the option specified by the *-icc2_pre_route_script* option of the *set_implement_options* command.

`-icc2_eco_route_script`

Removes the option specified by the *-icc2_eco_route_script* option of the *set_implement_options* command.

`-icc2_pre_extract_script`

Removes the option specified by the *-icc2_pre_extract_script* option of the *set_implement_options* command.

`-insert_fillers`

Specifies to reset and stop standard filler cell insertion.

Description

The *reset_implement_options* command removes options specified by the *set_implement_options* command.

If this command is specified with options, only those *set_implement_options* settings are removed.

If this command is specified with no options, all *set_implement_options* settings are removed.

If *reset_implement_options* is invoked after *check_eco* in the PrimeECO flow, it ends the IC Compiler II process.

For more information, see the man pages for the *set_implement_options* and *check_eco* commands.

See Also

- check_eco
- check_routes
- implement_eco
- report_implement_options
- save_block
- set_eco_options
- set_implement_options
- write_implement_changes

Feedback

reset_memory_max_power

Reset the worst-case max power for memory cells for each of instances.

Syntax

```
int reset_memory_max_power
[instance_list]
```

Data Types

instance_list	list

Arguments

`instance_list`

To specify the list of memory instances which needs to be reset.

Description

Resets the power set by "estimate_memory_max_power" command for the list of memory instances in the input list.

This command should only be run in average mode. update_power should be used to calculate power and report_power shows the power values is reset for memory cells after running the command.

Examples

The following example shows how to reset maximum memory power on a single cell in the design.

```
pwr_shell> reset_memory_max_power I_SINGLE_PORT
reset_memory_max_power I_SINGLE_PORT
VL flow: For cell: I_SINGLE_PORT Resetting memory cell max power to 0.0
1
```

See Also

- estimate_memory_max_power
- update_power
- report_power

reset_mode

Resets cell mode groups or design mode groups to the default state.

Feedback

Syntax

status *reset_mode*

```
[-type cell | design]
[-group group_list]
[instance_list]
```

Data Types

group_list	list
instance_list	list

Arguments

`-type cell | design`

Indicates the type of mode group to be set to default. This option has the following mutually-exclusive valid values: *design* and *cell*.

- The *cell* value specifies that cell mode groups are to be set to default. Cell mode groups are defined on library cells in the library.
- The *design* value specifies that design mode groups are to be set to default.

Design mode groups are user specified using the *define_design_mode_group* command. If the *-type* option is omitted from the command, this is equivalent to specifying *-type cell*.

`-group group_list`

Specifies a list of mode groups, each of which is to be reset to its default setting. If the *-type* option has a cell value, the *group_list* option must contain only cell mode groups. If the *-type* option has a design value, the *group_list* option must contain only design mode groups.

instance_list

Specifies a list of instances for which the specified cell mode groups are to be set to default. If no instance list is specified, all moded cells are considered. This list must only be included with the *-type cell* option. No error occurs if you reset the modes on a cell that has no modes.

Description

Resets mode groups to the default state of all modes enabled. Cell mode groups are defined in the library. Each library cell can have a set of cell mode groups. Each of these cell mode groups can have two or more cell modes. Each of these cell modes can be mapped to a set of timing arcs of the library cell.

Feedback

When a cell mode group is set to default for a given instance of the library cell all of its modes are enabled for that cell. All timing arcs of the enabled modes also get enabled with respect to modes for that cell.

Cell mode groups can have different states due to the *set_mode -type cell*, *set_mode -type design*, and conditional evaluation. When conflict arises the following precedence rule is employed:

- **set_mode -type cell**
- **set_mode -type_design**
- Evaluation of mode conditions

To reset the state of the cell mode group due to the *set_mode -type cell* command, use the *reset_mode -type cell* command.

To reset the state of the cell mode group due to the *set_mode -type design* command, use either the *reset_mode -type design* or *remove_design_mode* command.

To reset the state of the cell mode group due to conditional evaluation, use the *remove_case_analysis* command or edit the Verilog netlist to remove logical constants.

You define design modes and design mode groups using the *define_design_group* and *map_design_mode* commands. Design modes can be mapped to a set of cell modes, set of paths, or both cell modes and paths. When a design mode group is reset to default all design modes of that group are enabled. When a design mode is enabled all paths mapped to the design mode are reset and all previous setting of cell modes mapped to the design modes are removed.

Examples

In the following example, we have two cell mode groups RW and SH defined on cell Uram1/D0. The first command sets READ as the active mode for RW and EARLY as the active mode for SH.

```
pt_shell> set_mode {EARLY READ} Uram1/D0
```

The following command can be employed to reset the cell mode groups to default.

```
pt_shell> reset_mode -group {RW SH} Uram1/D0
```

To reset all cell mode group in Uram1/DO use the following command

```
pt_shell> reset_mode Uram1/D0
```

To reset all cell mode groups in the design use

```
pt_shell> reset_mode
```

In the following example, the first command defines two design modes, DM1 and DM2. (Since no mode group name was specified, the mode group is named "default".) The second command specifies that for design mode DM1, the READ cell mode is active for the RAM instance Uram1. The third command specifies that for design mode DM2, the WRITE cell mode is active for the RAM instance Uram1. The fourth command maps all paths ending at Uram1/D0 to the design mode DM2. The fifth command sets the design_mode DM1. The sixth command sets READ as the active cell mode and disables the WRITE cell mode.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode DM1 READ Uram1
pt_shell> map_design_mode DM2 WRITE Uram1
pt_shell> map_design_mode DM2 -to Uram1/D0
pt_shell> set_mode -type design DM1
pt_shell> set_mode -type cell READ Uram1
```

The following example resets the design mode group "default".

```
pt_shell> reset_mode -type design -group default
```

This command enables the design_mode DM2 resulting in the resetting of all paths mapped to Uram1/D0. However the cell mode WRITE on Uram1/D0 does not get enabled since it was also disabled by command 5 above. To reset the cell mode group to default use

```
pt_shell> reset_mode -type cell Uram1
```

The following command is equivalent to the previous command

```
pt_shell> reset_mode Uram1
```

See Also

- define_design_mode_group
- remove_design_mode
- report_mode
- map_design_mode
- set_mode
- set_case_analysis

---

reset_multi_input_switching_coefficient

Resets user-specified multi-input switching (MIS) coefficients.

Feedback

Syntax

int *reset_multi_input_switching_coefficient*

```
-all
object_list
```

Data Types

```
object_list          list
```

Arguments

`-all`

Resets all the library cells specified for multi-input switching analysis. You cannot use this option together with the *object_list* option.

`object_list`

Specifies a list of library cells to be reset. You cannot use this option together with the *-all* option.

Description

This command resets the user-specified base coefficients for multi-input switching (MIS) analysis. If you reset the coefficient of a library cell, then the tool does not perform multi-input switching analysis on that library cell.

Examples

The following example resets the base coefficients of library cell AND2 in the library MY_LIB:

```
pt_shell> reset_multi_input_switching_coefficient [get_lib_cells
 MY_LIB/AND2]
```

See Also

- set_multi_input_switching_coefficient
- report_multi_input_switching_coefficient
- si_enable_multi_input_switching_analysis
- si_enable_multi_input_switching_timing_window_filter

---

reset_noise_parameters

Resets the noise analysis parameters for the current design.

Feedback

Syntax

```
int reset_noise_parameters
```

Arguments

None.

Description

The *reset_noise_parameters* command resets the parameters that are considered during the noise analysis. If this command is used, the default settings are used for the noise analysis: beyond the rail analysis is ignored, arrival times of aggressors are considered and propagation of noise is disabled.

Examples

The following example resets the default noise parameters:

```
pt_shell> reset_noise_parameters
```

See Also

- report_noise
- report_noise_parameters
- set_noise_parameters
- update_noise

---

reset_ocvm_table_group

Resets the association between an AOCV or POCV named table group or a based POCV or AOCV derating group and a list of design objects, where an association has been performed earlier using the *set_ocvm_table_group* command.

Syntax

status *reset_ocvm_table_group*

```
-type aocvm | pocvm
[-table_type coefficient | distance]
[object_list]
```

Data Types

```
object_list     list
```

Feedback

Arguments

`-type aocvm | pocvm`

Specifies AOCV or POCV tables.

`-table_type coefficient | distance`

Specifies coefficient or distance-based POCV tables. The default is *coefficient*. Do not use this option for AOCV.

`object_list`

Specifies a list of design objects.

Description

The *reset_ocvm_table_group* command removes the association between a named AOCV or POCV table set and a hierarchical instance or between a lib_cell and a AOCV or POCV derating group, where such an association has been specified previously using the *set_ocvm_table_group* command.

Examples

In the following example, the *set_ocvm_table_group* command specifies an association between the hierarchical cell H1 and the named POCV coefficient table set A1. The *reset_ocvm_table_group* command that follows removes this association. Cells (or nets) fully enclosed within H1 then derive their POCV coefficients from a higher-level hierarchical cell that fully encloses H1 and has a named POCV coefficient table set associated with it. If no such hierarchical cell is found, the unnamed POCV coefficient table set is used.

```
pt_shell> set_ocvm_table_group -type pocvm -table_type coefficient A1
 [get_cells H1]

pt_shell> reset_ocvm_table_group -type pocvm -table_type coefficient
 [get_cells H1]
```

See Also

- get_timing_paths
- report_ocvm
- report_timing
- set_ocvm_table_group

---

reset_parasitics_range

Resets the parasitic range factors previously set by the set_parasitics_range command.

Feedback

Syntax

status *reset_parasitics_range*

Description

The *reset_parasitics_range* command resets the two factors of parasitic resistance range and two factors of parasitic capacitance range previously set by the set_parasitics_range command.

Examples

The following example resets all parasitic ranges.

```
pt_shell> reset_parasitics_Range

1
```

See Also

- read_parasitics
- set_parasitics_range

---

reset_path

Resets specified paths to single-cycle behavior.

Syntax

status *reset_path*

```
[-setup | -hold]
[-rise | -fall]
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
```

Data Types

```
from_list               list
rise_from_list          list
fall_from_list          list
through_list            list
rise_through_list       list
fall_through_list       list
```

Feedback

```
to_list                  list
rise_to_list             list
fall_to_list             list
```

Arguments

`-setup`

Indicates that only setup (maximum delay) evaluation is reset to its default, single-cycle behavior. If neither the *-setup* nor *-hold* options is specified, both setup and hold checking are reset to single-cycle.

`-hold`

Indicates that only hold (minimum delay) evaluation is reset to its default, single-cycle behavior. If neither the *-setup* nor *-hold* options is specified, both setup and hold checking are reset to single-cycle.

`-rise`

Indicates that only rising path delays are reset to single-cycle behavior. If neither the *-rise* nor *-fall* options is specified, both rising and falling delays are reset to single-cycle.

`-fall`

Indicates that only falling path delays are reset to single-cycle behavior. If neither the *-rise* nor *-fall* options is specified, both rising and falling delays are reset to single-cycle.

`-from` *from_list*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected. You can use only one of the following options: *-from*, *-rise_from*, or *-fall_from*.

`-rise_from` *rise_from_list*

Same as the *-from* option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: *-from*, *-rise_from*, or *-fall_from*.

`-fall_from` *fall_from_list*

Same as the *-from* option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by

the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: *-from*, *-rise_from*, or *-fall_from*.

`-through` *`through_list`*

Specifies a list of pins, ports, and nets through which the paths must pass that are reset. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you omit the *-through* option, all timing paths specified using the *-from* and *-to* options are affected.

You can use multiple *-through*, *-rise_through*, and *-fall_through* options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one *-through*, *-rise_through*, or *-fall_through* option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

`-rise_through` *`rise_through_list`*

This option is similar to the *-through* option, but applies only to paths with a rising transition at the through points.

`-fall_through` *`fall_through_list`*

This option is similar to the *-through* option, but applies only to paths with a rising transition at the through points.

`-to` *`to_list`*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of the following options: *-to*, *-rise_to*, or *-fall_to*.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but

only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: *-to*, *-rise_to*, or *-fall_to*.

`-fall_to` *`fall_to_list`*

Same as the *-to* option, but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the following options: *-to*, *-rise_to*, or *-fall_to*.

Description

The command restores designated timing paths in the current design to the default single-cycle behavior. Use the *reset_path* command to undo the effect of the *set_multicycle_path*, *set_false_path*, *set_max_delay*, and *set_min_delay* commands. Attributes set by these commands are removed by the *reset_path*. command.

Note that a general *reset_path* command removes the effect of matching general or specific point-to-point exception commands, if there is a matching object in the path specification of the original exception-setting commands. For example, the following command

```
reset_path -from {CLK}
```

resets more specific matching exceptions for the object CLK, such as

```
set_false_path -from {CLK} -to {d/Z}
set_false_path -from {CLK} -to {g/Z}
```

If you do not specify either the *-setup* or *-hold* option, the default behavior is restored to both. The default is a setup relation of one cycle and a hold relation of zero cycles, so that hold is checked one active edge before the setup data at the destination register.

Setup and hold information is different for rising and falling transitions. In most cases, these are reset together. Use the *-rise* or *-fall* options to reset only one or the other. To disable setup or hold calculations for paths, use the *set_false_path* command.

The general and specific constraint options (for example, *-through* and *-rise_through* or *-fall_through*) are treated as different qualifiers. That is, if you issue the *reset_path -fall_through* variable, only those constraints are removed that were set with the *-fall_through* option; any that were set with the *-through* option are not removed. The same applies to the general and specific *-to* and *-from* options. For an example, see the EXAMPLES section.

Feedback

Examples

The following command resets the timing relation between ff1/CP and ff2/D to the default single-cycle behavior.

```
pt_shell> reset_path -from { ff1/CP } -to { ff2/D }
```

The following command resets only the rising delay to single cycle for all paths ending at latch2d.

```
pt_shell> reset_path -rise -to latch2d
```

The following example first sets a specific and a general point-to-point exception, then removes the exceptions.

```
pt_shell> set_multicycle_path 2 -from A -to Z
pt_shell> set_max_delay 15 -to Z
pt_shell> reset_path -to Z
```

In following example, the *reset_path* command removes only the max delay constraints, because the *-to* and *-rise_to* options are treated as different qualifiers and do not match.

```
pt_shell> set_multicycle_path 2 -from A -to Z
pt_shell> set_max_delay 15 -rise_to Z
pt_shell> reset_path -rise_to Z
```

See Also

- current_design
- report_constraint
- reset_design
- set_false_path
- set_max_delay
- set_min_delay
- set_multicycle_path

---

reset_power_base_clock

Resets the user specified power_base_clock for a given set of objects.

Syntax

integer *reset_power_base_clock*

```
[object_list]
```

Feedback

Data Types

object_list list

Arguments

object_list

Specify a list of objects in the design. Only nets, pins and ports are supported.

Description

This command resets any user specified power_base_clock on given set of objects. It also resets the activities of the concerned objects annd invalidates the power data on the entire design .

User will need to specifiy activity on the concerned objects and rerun update_power.

---

reset_power_budget

Resets power budget set on the hierarchical instances in the design.

Syntax

int *reset_power_budget*

Description

This command resets power budget set on the hierarchical instances in the design.

Examples

In the following example, power budgets have been set on the following hierarchies and their rail.

```
pt_shell> set_power_budget 8.513e-05 -cell four_mini_inst/two_mini_1/a
pt_shell> set_power_budget 2.338e-04 -cell four_mini_inst/two_mini_1
 -rails four_mini_inst/VDD_MINI_1
pt_shell> set_power_budget 9.354e-05 -cell four_mini_inst/two_mini_1/a
 -rails four_mini_inst/VDD_MINI_1
pt_shell> report_power_budget
****************************************
Report : power budget
Design : top
****************************************

    Cell Name                     Rail                            Power
 Budget        Scale Factor
 ------------------------------------------------------------------------
------------------
    four_mini_inst/two_mini_1     four_mini_inst/VDD_MINI_1
 2.104405e-04        2.50
```

```
    four_mini_inst/two_mini_1/a    --
 8.513000e-05        1.50
    four_mini_inst/two_mini_1/a    four_mini_inst/VDD_MINI_1
 9.354000e-05        2.00
1
```

The following command resets power budgets set on the above hierarchical instances and their rails.

```
pt_shell> reset_power_derate
pt_shell> report_power_budget
****************************************
Report : power budget
Design : top
****************************************

    Cell Name                     Rail                            Power
 Budget        Scale Factor
 ------------------------------------------------------------------------
------------------
1
```

See Also

- report_power_derate
- set_power_derate

---

reset_power_delay_shifted_event_analysis_options

Resets the options for delay shifted event analysis.

Syntax

integer *reset_power_delay_shifted_event_analysis_options*

Description

This command is used when *power_enable_delay_shifted_event_analysis* is set to true. The command is used to reset the delay shifted event analysis options set by '*set_power_delay_shifted_event_analysis_options*.

See Also

- power_enable_delay_shifted_event_analysis
- set_power_delay_shifted_event_analysis_options
- report_power_delay_shifted_event_analysis_options

Feedback

reset_power_derate

Resets power derating factors set either on a design or on a specified list of instances (cells, library cells, nets or pins).

Syntax

int *reset_power_derate*

```
[-groups group_name_list]
[-rails rail_list]
[-pg_pins pg_pin_list]
object_list
```

Data Types

```
group_name_list     list
rail_list           list
pg_pin_list         list
object_list         list
```

Arguments

`-groups group_name_list`

Specifies the power group or groups in which the derate factors of all the cells are reset.

`-rails rail_list`

Specifies a list of power supply nets in which the derate factors of all the objects are reset.

`-pg_pins pg_pin_list`

Specifies a list of pg_pins on which the derate factors for all the objects are reset.

`object_list`

Specifies current design or a list of cells or library cells on which the power derating factors are reset to 1.0.

Description

Invoke this command with no arguments to reset all derate factors set on the design (to the default of 1.0). This includes both derating factors set globally on the design and those set on specific instances in the design. If the command is called with a list of objects, only the power derating factors on the specified objects are reset. After a *reset_power_derate* command is applied to an object, the power derating factors of all power components, including switching, internal, and leakage power are reset. If the *object_list* variable

contains the current design, then only global power derating factors are reset and instance specific power derating factors remain untouched.

If the *object_list* variable contains any hierarchical cells, then all cells within that hierarchical cell and also all cells of lower hierarchies will have their inherited power derating factors updated accordingly. Note that power derating factors set specifically on an object are not overwritten by a set or reset command on its parent hierarchical cell. To reset the power derating factors set specifically on the child cells, include them in the object list of the command.

The *-groups* options allows to reset the derate factors of cells only in the specified groups.

Heterogeneous collections of objects can be combined and passed to this command.

If the *object_list* variable contains any hierarchical cells, then all cells within that hierarchical cell and also cells of all lower hierarchies will have their inherited derate factors updated accordingly. Note that derate factors set specifically on an object are not overwritten by a set or reset command on its parent hierarchical cell.

The command resets x-transition power derate factors on net or on pin objects if nets or pins are passed in object_list, having x-transition power derate factors.

The command resets pin switching power derate factors on pin objects.

Examples

The following command resets both power derating factors set globally on the design and those set on specific instances in the design.

```
pt_shell> reset_power_derate
```

The following command resets the power derating factors set globally on the design MY_DESIGN only.

```
pt_shell> reset_power_derate [get_design MY_DESIGN]
```

The following example resets the power derating factors set on cell U1.

```
pt_shell> reset_power_derate [get_cell U1]
```

The following command resets the power derating factors set on all cells matching "*".

```
pt_shell> reset_power_derate [add_to_collection [get_cells *]
```

The following example resets the power derating factors set on all instances of library cell IV in the library MY_LIB.

```
pt_shell> reset_power_derate [get_lib_cells MY_LIB/IV]
```

Feedback

Assuming that the hierarchical cell H1 contains a cell U1, then the following command resets the derate factors set on U1. As a result, U1 inherits the derate factors that are set on H1.

```
pt_shell> reset_power_derate [get_cells H1/U1]
```

Assuming that the hierarchical cell H1 also contains a hierarchical cell H2, then the following command resets the power derate factors set on H2. As a result, H2 and all of its cells inherit the power derating factors that are set on H1.

```
pt_shell> reset_power_derate [get_cells H1/H2]
```

Following command will reset the x-transition power derate factors on the given list of nets, assuming nets have x-transition derate factors.

```
pt_shell> reset_power_derate {U0/n5 U0/CP U0/Q U0/n6}
```

See Also

- report_power
- report_power_derate
- set_power_derate

---

reset_pvt_explorer_condition

Resets PVT explorer condition for PrimeShield PVT Variation Exploration what-if analysis.

Syntax

string *reset_pvt_explorer_condition*

Description

The *reset_pvt_explorer_condition* command will reset PVT explorer condition specified by *set_pvt_explorer_condition* command, which is used in PrimeShield Power, voltage, and temperature (PVT) parameter variation and exploration.

define_sensitivity_lib_mapping is required in this feature. And *set ps_enable_pvt_explorer_analysis true* is required to use *reset_pvt_explorer_condition* command.

Examples

The following script enables Power, performance, area (PPA) what-if analysis, which enables you to change the design PVT parameters and quickly assess their PPA impact. It allows process sweet-spot exploration and technology option assessment, then reset it.

"one_cond.txt" file content as follows:
#index,pmos_lvt_vt,nmos_lvt_vt,pmos_lvt_ids0,nmos_lvt_ids0 1,0.020,0.020,0.15,0.15

The timing and power analysis results after *set_pvt_explorer_condition* will reflect the physical parameter shift defined in "one_cond.txt".

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_pvt_explorer_analysis true
set ps_pvt_explorer_pba_only_mode false
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]

set_pvt_explorer_condition one_cond.txt
update_timing -full
update_power
# timing and power here has impact from device parameter shift impact.

reset_pvt_explorer_condition
update_timing -full
update_power
# timing and power here is without impact from device parameter shift
 impact.
```

You must change the "one_cond.txt" file content to include the device model names and values, based on your PDK or side file. The "one_cond.txt" file includes the delta shift values. In the preceding example, the vt shift provided is +20mV and the idsat shift provided is +15%, from their respective nominal values.

Timing robustness what-if analysis enables you to perform path-based timing analysis with multiple what-if specifications. It allows you to identify paths susceptible to PVT changes, o provide ECO guidance to improve the design robustness.
The following script enables what-if analysis for timing path robustness, where "multiple_DoEs.txt" file content as follows to cover multiple what-if specifications:
#index,pmos_lvt_vt,nmos_lvt_vt,pmos_lvt_ids0,nmos_lvt_ids0 1,0.020,0.020,0.15,0.15
2,0.011,0.011,0.10,0.10 Then *reset_pvt_explorer_condition* then PBA paths afterwards will not trigger timing robustness what-if analysis.

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_pvt_explorer_analysis true
set ps_pvt_explorer_pba_only_mode true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]
update_timing -full  # baseline QoR
set paths [get_timing_paths ...]

set_pvt_explorer_condition multiple_DoEs.txt
#update original path collection with slack shift per DoE
```

Feedback

```
set paths_new [get_timing_paths -pba_mode path $paths]
#generate slack shift per DoE for all paths
puts [get_attr $paths_mew pvt_explorer_slack_shift]

reset_pvt_explorer_condition multiple_DoEs.txt
set paths_after_reset [get_timing_paths -pba_mode path $paths]
# paths_after_reset doesn't have pvt_explorer_slack_shift attribute and
 slack is same as "paths"
```

See Also

- define_sensitivity_lib_mapping
- report_pvt_explorer_condition
- set_pvt_explorer_condition
- ps_enable_spice2design_analysis
- ps_enable_pvt_explorer_analysis
- ps_pvt_explorer_pba_only_mode

---

reset_routing_rule

Resets non-default routing rule changes back to the pre-ECO values.

Syntax

integer *reset_routing_rule*

```
[-all]
[net_list]
```

Data Types

net_list	list

Arguments

`-all`

Resets all routing rule changes back to pre-ECO values.

net_list

Lists of nets to reset the routing rule change.

Description

This command resets routing rule changes performed by *set_routing_rule* command.

Feedback

Examples

The following example resets the routing rule change to a net named *n43* back to the pre-ECO values.

```
prompt> reset_routing_rule {n43}
```

See Also

- set_routing_rule
- report_routing_rules

---

reset_rtl_to_gate_name

Resets all the RTL-to-gate name mapping information.

Syntax

integer *reset_rtl_to_gate_name*

Arguments

None.

Description

The *reset_rtl_to_gate_name* command resets all the RTL-to-gate name mapping information stored previously using the *set_rtl_to_gate_name* and *unset_rtl_to_gate_name* commands.

Examples

The following example provides the resetting of the RTL-to-gate name mapping.

```
pt_shell> reset_rtl_to_gate_name
```

See Also

- set_rtl_to_gate_name
- unset_rtl_to_gate_name

---

reset_scale_parasitics

Resets the parasitic scaling factors previously set by the *scale_parasitics* command.

Syntax

status *reset_scale_parasitics*

```
[net_list]
```

Data Types

```
net_list        list
```

Arguments

net_list

Limits the resetting of parasitic scaling factors to only the listed nets. By default, the command resets all net-specific and global parasitic scaling settings. Using this option, only nets with net-specific scaling factors can be reset.

Description

The *reset_scale_parasitics* command resets the parasitic resistance and capacitance scaling factors previously set by the *scale_parasitics* command. By default, both the global and net-specific scaling factor settings are reset.

You can optionally restrict the resetting to only a specified list of nets that have net-specific scaling factor settings.

Examples

The following example resets all parasitic scaling factors.

```
pt_shell> reset_scale_parasitics
1
```

See Also

- read_parasitics
- report_scale_parasitics
- scale_parasitics

---

reset_sensitivity_lib_mapping

Resets the mapping between base library and sensitivity augmented library.

Syntax

string *reset_sensitivity_lib_mapping*

```
[libraries]
```

Data Types

```
libraries                list
```

Feedback

Arguments

libraries

Specifies a list of libraries to reset mapping of base library and sensitivity agumented library. If this option is omitted, side file for all of libraries will be reset.

Description

The *reset_sensitivity_lib_mapping* command resets mapping between base library and sensitivity augmented library, which is defined in *define_sensitivity_lib_mapping*.

Examples

The following script enables SPICE2Design flow, defines mapping between base library and sensitivity side-file, report the mapping, reset, then report again.

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_spice2design_analysis true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]

pt_shell> report_sensitivity_lib_mapping orig
  Lib             side_file
================================================
  orig            side.db
================================================
1

reset_sensitivity_lib_mapping
pt_shell> report_sensitivity_lib_mapping
  Lib             side_file
================================================
================================================
1
```

See Also

- define_sensitivity_lib_mapping
- report_sensitivity_lib_mapping
- ps_enable_spice2design_analysis
- ps_enable_pvt_explorer_analysis

Feedback

reset_switching_activity

Resets switching activity annotation on nets, pins, ports, and cells of the current design.

Syntax

int *reset_switching_activity*

```
[-static_probability]
[-toggle_rate]
[-state_condition state]
[-path_sources name_list]
[-no_hierarchy]
[object_list]
[-quiet]
```

Data Types

state	string
name_list	string
object_list	list

Arguments

`-static_probability`

Specifies the static probability value to reset for the given design object. The static probability value is internally reset to an invalid number.

`-toggle_rate`

Specifies the toggle rate (rise and fall, if applicable) and glitch rate (rise and fall, if applicable) to reset for the given design object. The toggle and glitch rate values are internally reset to invalid numbers.

`-state_condition` *state*

Specifies the state condition when resetting state-dependent toggle rate and glitch rates on pins or state-dependent static probabilities on cells. State-dependent toggle rate and glitch rate can be annotated when the internal power of the library cell pin is characterized with state-dependent power tables. State-dependent static probabilities can be annotated when the cell leakage power is characterized with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal/leakage power characterization. The state condition should be enclosed between " ". Moreover, if you want to reset switching activity information for the default state condition, the argument of the *-state* option should be "default". Note that the *-state_condition* option applies only to the leaf level cells.

`-path_sources` *`name_list`*

Specifies the path sources when resetting path-dependent toggle rate and glitch rate on pins. This is used when the library cell pin has path-dependent internal power. The path sources specified with this argument must be the same as those in the internal power characterization. When listing more than one pin in path sources, the pin names should be separated by a space and enclosed between quotes " ". For example, if the pins in path sources are A and B, the argument for the *-path* option looks like "A B". The *-path_sources* argument applies only to the leaf level cells. Note that if the *name_list* option is given using the *-path_sources* argument, the *state* option should also be provided using the *-state_condition* option.

`-no_hierarchy`

This option specifies that only the objects in the hierarchy of the list of instances, specified using the *object_list* option, is reset. If no instance list is specified, then the switching activity on the objects in the same hierarchy of the current instance is reset. If the *-no_hierarchy* option is not specified and the *object_list* option is also not given, the switching activity on all the objects in all the hierarchy of the current instance are reset. Note that if leaf level cells are given using the *object_list* option, the command internally assumes this option is true.

`object_list`

Specifies a list of nets, pins, ports, or instances in the current design on which the static probability, toggle rate, and glitch rate switching activity values are reset.

`-quiet`

Specifies quiet mode and suppresses warnings on design objects for which switching activity could not be reset.

Description

Use this command to reset design nets, ports, pins, and cells with the different kinds of switching activity. These include simple toggle rate, glitch rate, and static probability on nets, ports and pins; state-dependent and path-dependent toggle, and glitch rate on cell pins, and state-dependent static probabilities on cells. Toggle and glitch rates, and static probabilities are reset using the *-toggle_rate* and *-static_probability* options, respectively. You can make the toggle rate, glitch rate, and static probability state-dependent by specifying the *state* option with the *-state_condition* option. You can make the toggle rate and glitch rate path-dependent by specifying the path sources with the *-path_sources* option. The design objects that are reset with switching activity can be specified as a list of objects. You can reset only the objects in the same hierarchy by using the *-no_hierarchy* option.

Most of the above functionality and all the options apply when the *power_analysis_mode* variable is set to the *averaged* or *leakage_variation* option . When the*power_analysis_mode* variable is set to the *time_based* option, the *reset_switching_activity* command has a different meaning. In this mode, the *reset_switching_activity* command removes any VCD or other activity indicated previously with the *read_vcd* command. Because the *time_based* mode is not based on toggle rates, attempting to reset switching activity on individual objects has no effect.

Examples

The following example shows resetting toggle rate, glitch rate, and static probability values only on the current instance of the TOP_MULT design:

```
pt_shell> current_design TOP_MULT

pt_shell> reset_switching_activity -no_hierarchy

pt_shell> reset_switching_activity -no_hierarchy -toggle_rate
 -static_probability

pt_shell> reset_switching_activity -toggle_rate -no_hierarchy

pt_shell> reset_switching_activity -static_probability -no_hierarchy
```

The following example shows resetting the toggle rate, glitch rate, and static probability values on the whole TOP_MULT design:

```
pt_shell> current_design TOP_MULT

pt_shell> reset_switching_activity

pt_shell> reset_switching_activity -toggle_rate -static_probability

pt_shell> reset_switching_activity -toggle_rate

pt_shell> reset_switching_activity -static_probability
```

The following example shows resetting the toggle rate, glitch rate, and static probability values on the list of instances:

```
pt_shell> current_design TOP_MULT

pt_shell> reset_switching_activity {U1 U2}

pt_shell> reset_switching_activity -no_hierarchy {U1 U2}

pt_shell> reset_switching_activity -toggle_rate {U1 U2}

pt_shell> reset_switching_activity -static_probability {U1 U2}

pt_shell> reset_switching_activity -no_hierarchy -toggle_rate {U1 U2}
```

```
pt_shell> reset_switching_activity -no_hierarchy -static_probability
          {U1 U2}
```

The following *reset_switching_activity* command resets simple toggle rate value, glitch rate value, and static probability value on all design input ports:

```
pt_shell> reset_switching_activity -toggle_rate
         -static_probability [get_inputs]

pt_shell> reset_switching_activity [get_inputs]
```

The following example resets different switching activity values on design output ports:

```
pt_shell> reset_switching_activity -toggle_rate [get_outputs]

pt_shell> reset_switching_activity -static_probability
          [get_outputs]

pt_shell> reset_switching_activity -static_probability
         -toggle_rate [get_outputs]
```

The following example resets state-dependent static probabilities on the or1 cell:

```
pt_shell> reset_switching_activity -static_probability
         -state_condition "A & B" [get_cell or1]

pt_shell> reset_switching_activity -state_condition "A & B"
         [get_cell or1]

pt_shell> reset_switching_activity -static_probability
         -state_condition "A & ! B" [get_cell or1]

pt_shell> reset_switching_activity -static_probability
         -state_condition "! A & B" [get_cell or1]

pt_shell> reset_switching_activity -static_probability
         -state_condition "! A & ! B" [get_cell or1]

pt_shell> reset_switching_activity -static_probability
         -state_condition "default" [get_cell or1]
```

The following example resets simple and path sources, dependent toggle rates, and glitch rates on the output pin Y of the cell xor1:

```
pt_shell> reset_switching_activity -toggle_rate [get_pin xor1/Y]

pt_shell> reset_switching_activity -toggle_rate -path_sources "A"
         -state_condition "B" [get_pin xor1/Y]

pt_shell> reset_switching_activity -path_sources "A"
         -state_condition "B" [get_pin xor1/Y]
```

Feedback

```
pt_shell> reset_switching_activity -toggle_rate -path_sources "B"
        -state_condition "A" [get_pin xor1/Y]

pt_shell> reset_switching_activity -toggle_rate -path_sources "A B"
        -state_condition "default" [get_pin xor1/Y]
```

See Also

- get_switching_activity
- read_saif
- report_power
- report_switching_activity
- set_switching_activity

---

reset_timing_derate

Resets the derating previously set by the *set_timing_derate* command.

Syntax

status *reset_timing_derate*

```
[-hierarchical_net_delay]
[-scalar] [-variation]
[-aocvm_guardband] [-pocvm_guardband] [-pocvm_coefficient_scale_factor]
[-pocvm_subtract_sigma_factor_from_nominal]
[-increment]
[-min_period]
[-min_pulse_width]
object_list
```

Data Types

```
object_list      list
```

Arguments

`-hierarchical_net_delay`

Resets net delay derating (as well as cell delay derating) previously set by the *set_timing_derate* command for the hierarchical cells specified in the *object_list*. Without the *-hierarchical_net_delay* option, the *reset_timing_derate ... [get_cells ...]* command resets only the cell delay (not net delay) derating in the hierarchical cells.

`-scalar`

Resets derating previously set by the *set_timing_derate -scalar* ... command. This option should be used only with the PrimeTime VX tool.

`-variation`

Resets derating previously set by the *set_timing_derate -variation* ... command. This option should be used only with the PrimeTime VX tool.

`-aocvm_guardband`

Resets derating previously set by the *set_timing_derate -aocvm_guardband* ... command.

`-pocvm_guardband`

Resets derating previously set by the *set_timing_derate -pocvm_guardband* ... command.

`-pocvm_coefficient_scale_factor`

Resets derating previously set by the *set_timing_derate -pocvm_coefficient_scale_factor* ... command.

`-pocvm_subtract_sigma_factor_from_nominal`

Resets derating previously set by the *set_timing_derate -pocvm_subtract_sigma_factor_from_nominal* ... command.

`-increment`

Resets incremental derating adjustments previously set by the *set_timing_derate -increment* ... command.

`-min_period`

Resets min period derating adjustments previously set by the *set_timing_derate -min_period* ... command.

`-min_pulse_width`

Resets min pulse width derating adjustments previously set by the *set_timing_derate -min_pulse_width* ... command.

`object_list`

Resets the derating previously set for the given list or collection of cells, library cells, or nets.

Description

The *reset_timing_derate* command resets the derating previously set by the *set_timing_derate* command, which effectively changes derating factors back to their

original default values (1.0 for delay or constraint derating, or 0.0 for *-increment* derating adjustments).

To reset all derating, including both global and object-specific derating settings, use the *reset_timing_derate* command by itself, without any options.

To reset only some types of derating, use the same scope-setting options as the original *set_timing_derate* command:

```
-aocvm_guardband
 -pocvm_guardband
 -pocvm_coefficient_scale_factor
 -increment
 object_list
```

By default, using the *reset_timing_derate* command and specifying a hierarchical cell in the *object_list* causes only the cell delay derating to be reset. To also reset the net derating, use the *-hierarchical_net_delay* option in the command.

If the *object_list* option contains a design, only global derating factors are reset, and object-specific derating factors are retained. For example, to reset all global derating but keep derating set on specific cells and nets, use the following command:

```
pt_shell> reset_timing_derate [current_design]
```

To remove the derating on a subset of an existing derating setting, do not use the *reset_timing_derate* command. Instead, set a derating factor of 1.0 on the subset using another *set_timing_derate*, specifying the more narrow condition (*-net_delay*, *-rise*, and so on) or more specific *object_list*.

Examples

The following command resets all derating, including both global and instance-specific settings.

```
pt_shell> reset_timing_derate
```

The following command resets all incremental derating adjustments previously set the *set_timing_derate -increment* ... commands, including both global and instance-specific settings:

```
pt_shell> reset_timing_derate -increment
```

The following command resets all global derating set on the design MY_DESIGN, without affecting the derating set on specific objects in the design (cells, library cells, and nets).

```
pt_shell> reset_timing_derate [get_design MY_DESIGN]
```

The following example resets all derating previously set on cell U1.

```
pt_shell> reset_timing_derate [get_cell U1]
```

Feedback

The following example resets all derating previously set on net n123.

```
pt_shell> reset_timing_derate [get_net n123]
```

The following command resets all derating previously set on cells matching "U2*" and on all nets matching "n1*".

```
pt_shell> reset_timing_derate \\
  [add_to_collection [get_cells U2*] [get_nets n1*]]
```

The following example resets derating previously set on instances of library cell IV1A from the library MY_LIB.

```
pt_shell> reset_timing_derate [get_lib_cells MY_LIB/IV1A]
```

The following command resets the derating previously set on leaf-level cell H1/U1. As a result, cell U1 inherits the derating set on its parent cell, H1.

```
pt_shell> reset_timing_derate [get_cells H1/U1]
```

The following command resets the cell delay derating previously set on hierarchical cell H1/H2. As a result, cell H2 and all of its lower-level cells inherit the cell delay derating set on their parent (or grandparent) cell, H1.

```
pt_shell> reset_timing_derate [get_cells H1/H2]
```

The following command resets the cell *and net* delay derating previously set on hierarchical cell H1/H2. As a result, cell H2 and all of its lower-level cells inherit the cell *and net* delay derating set on their parent (or grandparent) cell, H1.

```
pt_shell> reset_timing_derate -hierarchical_net_delay [get_cells H1/H2]
```

See Also

- set_timing_derate
- report_design
- report_timing_derate

---

reset_vsa_margin_table

Resets the margin table previously set by the *set_vsa_margin_table* command.

Syntax

status *reset_vsa_margin_table*

```
[-cell lib_cell]
```

Feedback

DATA TYPES

lib_cell string

Arguments

`-cell lib_cell`

Specifies the library macro cell instance for which you want to remove the VSA margin table.

Description

The *reset_vsa_marign_table* command resets the margin table, for a specified macro cell instance, previously set by the *set_vsa_margin_table* command.

To reset the margin table for all library macro cells, use the *reset_vsa_marign_table* command by itself, without any option.

Examples

The following command resets the VSA margin table of the library cell, *MACRO*:

```
pt_shell> reset_vsa_margin_table -cell MACRO
```

The following command reset the VSA margin table of all library macro cells:

```
pt_shell> reset_vsa_margin_table
```

See Also

- set_vsa_margin_table

---

restore_session

Restore a PrimeTime session from a directory saved by the *save_session* command.

Syntax

int *restore_session*

```
[-name session_name]
[-partitions instances_or_design_names]
[-program_options save_version | restore_version]
dir_name
```

Data Types

```
session_name          string
dir_name              string
```

Feedback

Arguments

`-name session_name`

This option, if specified when restoring a timing path session, allows assigning a name for the restored path session.

`-partitions instances_or_design_names`

Specifies the block instances to be restored in the distributed analysis flow along with the top-level partition. By default, all partitions are restored. Using the *-partitions* option overrides the default behavior, causing only the listed blocks and top level to be restored.

You can also provide block design names instead of instance names in the list. In this case, any partition whose design name matches the design names in the list will be restored.

`-program_options save_version | restore_version`

This option, if specified will load program settings of save_session or restore_session PrimeTime version. By default, it is save_version.

`dir_name`

Specifies the name of a directory to read the session information from.

Description

Use this command to read a directory that was written by the *save_session* command and restore a PrimeTime session to the same state as the session where the *save_session* command was issued.

When a version-specific session is restored, ensure the version of PrimeTime used to restore a session with the *restore_session* command is the same version used when saving the session with the *save_session* command. Locate the version used to save the session from the README file located in the session directory.

When a version-compatible session is restored, an information message indicates the original version:

```
Information: Restoring the version compatible session saved with
version 'Q-2019.12-SP4'. (SR-048)
```

If there is existing design or library data in memory, it is removed before restoring the session.

The command first tries to grab all licenses that were present at the time of the *save_session* command. If this fails, the command fails.

The design data is restored from data in the session directory. The *restore_session* command reads the library files that are needed to restore the session.

The restore directory contains a text file named 'lib_map'. This file contains the path and leaf name for each logic library needed to restore the session. If necessary, you can edit the path names to reflect the current environment. The leaf names of those files cannot be changed. The logic libraries are assumed to be the same as those used when the *save_session* command was issued.

When a saved session is restored, any variable that existed in the saved image is restored to its saved value. However, any variables that existed in an analysis before the *restore_session* command is issued, which were not defined in the saved image, remain unchanged.

An arbitrary collection of *timing_path* objects generated in a PrimeTime session can be saved using the *-only_timing_paths* option of the *save_session* command. In this case, only the *timing_path* collection and supporting data are saved, not the full design or timing information. Subsequently, multiple path sessions can be loaded into memory in a regular PrimeTime environment using the *restore_session* command. If a design has been instantiated prior to restore of a path session, then an error is generated and the path collection is not loaded. By default, the name of the path collection restored is the same as the name of the session directory and each *timing_path* object within the restored collection inherits the path session name as a read-only application attribute *session_name*. The default path session name can be overwritten by using the *-name* option.

If the session is saved from a distributed analysis flow, then you have the option to restore all or subset of partitions using the *-partitions* option. But, the top-level partition is always restored.

Examples

This example reads a saved PrimeTime session from a directory named state1.

```
pt_shell> restore_session state1
```

The following example restores a collection of timing paths:

```
pt_shell> restore_session path_session_dir -name mypaths
```

The following three examples shows different ways to restore a distributed analysis session.

The first example restores a complete distributed analysis session, including the distributed manager, and recreates the distributed database. It starts the same number of resources that were used to create the session.

```
pt_shell> restore_session da_session
```

The second example restores three partitions: nlkA/hier1, blkB/hier2, and top-level partition. It also restores the distributed manager and creates a distributed system.

```
pt_shell> restore_session -partitions {blkA/hier1 blkB/hier2} da_session
```

The third example only restores a single partition as a standalone PrimeTime session. Here, the tool restores only partition1 of da_session.

```
pt_shell> restore_session da_session/partition1
```

The following example load program settings of restore PT version:

```
pt_shell> restore_session session_dir -program_options restore_version
```

See Also

- save_session

---

restore_timing_paths

Restores a collection of timing paths saved by the *save_timing_paths* command.

Syntax

collection *restore_timing_paths*

```
        dir_name
```

Data Types

dir_name	string

Arguments

dir_name

Specifies the name of the directory from which to read the timing path collection saved by the *save_timing_paths* command.

Description

Use this command to restore a timing path collection stored by the *save_timing_paths* command. The restored path collection is just like a collection created by the *get_timing_paths* command.

The design, constraints, and version of the PrimeTime tool must be exactly the same as when the path collection was saved.

Feedback

Examples

The following example saves a timing path collection named my_paths1 into a directory named paths1, and later restores the path collection:

```
pt_shell> set my_paths [get_timing_paths -nworst 20 -max_paths 500]
 ...
pt_shell> save_timing_paths $my_paths -output paths1
 ...

New session:

pt_shell> set my_paths [restore_timing_paths paths1]
 ...
pt_shell> report_timing $my_paths
 ...
```

The following example saves a timing path collection directly into a directory named paths2, and later restores the path collection and reports its timing:

```
pt_shell> save_timing_paths [get_timing_paths -delay_type min] -output
 paths2
 ...

New session:

pt_shell> report_timing [restore_timing_paths paths2]
 ...
```

See Also

- get_timing_paths
- save_timing_paths

---

rotate_objects

Rotates the specified objects/ cells

Syntax

status *rotate_objects*

```
-orient orient
-angle angle
[-anchor center | ll | lr | ul | ur]
[-pivot {x y}]
[-force]
[-simple]
[object_list]
```

Feedback

Data Types

```
orient       string
angle        string
x            float
y            float
object_list  collection or selection
```

Arguments

object_list

Collection or selection set that contains objects to rotate. If this option is not specified, global selection set is used.

-orient *orient*

Specifies the cell orientation.

The valid values are:

```
N,  W,  S,  E,
FN, FS, FE, FW,
NW, NE, EN, ES,
SE, SW, WN, WS
```

-angle *angle*

Specifies the rotation angle of the objects.

The valid values are:

```
0, 90, 180, 270,
CW90, CW180, CW270, CCW90, CCW180, CCW270
FLIPX, FLIPY
```

The rotation value definitions are:

```
CW90    is 90 degrees clockwise
CW180   and CCW180 are 180 degrees
CW270   is 270 degrees clockwise
CCW90   is 90 degrees counterclockwise
CCW270  is 270 degrees counterclockwise
FLIPX   is the reflection about the x-axis
FLIPY   is the reflection about the y-axis
```

The following values are synonymous:

```
0,   R0
CW90,  90
CW180, 180
CW270, 270,
CCW90,  R90
CCW180, R180
```

CCW270, R270,
FLIPX, MY
FLIPY, MX

`-anchor center | ll | lr | ul | ur`

Specifies anchor point for rotation.

The anchor values are:

center is center
ll is lower left
lr is lower right
ul is upper left
ur is upper right

`-pivot {x y}`

Specifies the rotation pivot point.

The option cannot be specified together with the *-anchor* option.

`-force`

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

`-simple`

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

Description

This command rotates one or more specified objects by given angle skipping all fixed objects. It can also set selected cell orientation to given value.

Snapping is done automatically using the global snap settings.

Examples

The following example rotates all selected objects by 90 degrees.

```
prompt> rotate_objects -angle 90
```

The alternative syntax uses a collection to specify objects.

```
prompt> rotate_objects [get_selection] -angle 90
```

Feedback

See Also

- change_selection
- get_selection
- move_objects
- snap_objects

S

save_block

Saves the current NDM view of a block as a new block.

Syntax

```
status save_block
-as block_name
```

Data Types

```
block_name  string
```

Arguments

`-as block_name`

Specifies the block using a plain block name. Do not include a library name, label name, or view name.

Description

This command saves the current design block, which has been modified by ECO operations, into an IC Compiler II reference design library.

You must use the *-as* argument to specify the destination block name. Specify a plain block name, without any library, label, or view name, and without colon or slash characters.

The block is saved in the library specified by the *set_eco_options -physical_icc2_lib* option command, without a label name, and with the view name ".design".

Examples

The following command saves the block as "CPU_after_eco" in the library specified by the *set_eco_options -physical_icc2_lib* option command.

Feedback

```
prompt> save_block -as CPU_after_eco
Information: Saving 'design.nlib:ecoBlock_pid19.design' to
'design.nlib:CPU_after_eco.design'. (DES-028)
1
```

See Also

- check_eco
- check_routes
- implement_eco
- report_implement_options
- set_eco_options
- set_implement_options
- write_implement_changes

---

save_drc_error_data

Saves an external DRC error data object to disk.

Syntax

status *save_drc_error_data*

```
drc_error_data
[-as file_name]
```

Data Types

```
drc_error_data      collection
file_name           string
```

Arguments

drc_error_data

Specifies the collection of external physical DRC error data objects to save.

-as *file_name*

Specifies a new file name for the external error data file.

If you do not specify this option, the error data file is saved with its original name.

Feedback

Description

The *save_drc_error_data* command saves the specified external DRC error data object to disk.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

Examples

The following example opens the external error data file named my_design_dppinassgn.err, removes fixed errors, and then saves and closes the file.

```
prompt> set data [open_drc_error_data -file_name
 my_design_dppinassgn.err]
{my_design_dppinassgn.err}
prompt> remove_drc_errors -error_data $data \\
   [get_drc_errors -error_data $data -filter {status==fixed}]
prompt> save_drc_error_data $data
1
prompt> close_drc_error_data $data
1
```

See Also

- close_drc_error_data
- create_drc_error_data
- get_drc_error_data
- open_drc_error_data
- remove_drc_error_data

---

save_qtm_model

Saves the current quick timing model (QTM) description.

Syntax

string *save_qtm_model*

```
[-format db | lib]
[-output file_name]
[-library_cell]
```

Data Types

```
file_name     string
```

Feedback

Arguments

`-format db | lib`

Specifies the output format to be used. You can specify *db* (the default) or *lib*.

`-output` *`file_name`*

Specifies the name of the output file. By default, the tool writes the db file to *model_name*_lib.db. If you use the *-output* option, the tool writes the db file to *output_file*_lib.db file.

`-library_cell`

Writes the model as a library cell instead of a wrapper design and core library cell. If you use this option, no wrapper design is written, and the model in the *output_file*_lib.db file is named *model_name*.

If you do not specify this option, a wrapper design is written to the *output_file*.db file, and the model in the *output_file*_lib.db file is named *model_name*_core.

Description

This command saves the quick timing model and indicates that the model definition is complete. You must issue all quick timing model commands between the *create_qtm_model* and *save_qtm_model* commands.

To display information about the current quick timing model, use the *report_qtm_model* command.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following example saves a wrapper and core quick timing model as two files:

```
pt_shell> save_qtm_model -format db -output myfile
```

The preceding command generates these files:

- myfile_lib.db - Contains the *model_name*_core library cell
- myfile.db - Contains the *model_name* wrapper design

The following command saves the quick timing model as the *model_name* library cell in the myfile_lib.db file:

```
pt_shell> save_qtm_model -format db -output myfile -library_cell
```

The following example saves the quick timing model as a library cell in .lib format and uses the string "current" for the base file name.

```
pt_shell> save_qtm_model -format lib -output current -library_cell
```

Feedback

See Also

- create_qtm_model
- report_qtm_model

---

save_session

Saves the data of a PrimeTime session in a named directory, which can later be restored with the *restore_session* command.

Syntax

int *save_session*

```
[-include incl_list]
[-only_used_libraries]
[-only_timing_paths path_collection]
[-incremental]
[-name session_name]
[-disable_common_data_sharing]
[-version specific | compatible | both]
dir_name
```

Data Types

```
incl_list               list
path_collection         collection
session_name            string
dir_name                string
```

Arguments

`-version specific | compatible | both`

Specifies whether to save a version-specific or version-compatible session (or both). Allowed values are:

- *specific* Saves the session in a form that can be restored into the same PrimeTime release. Although this is the default, this keyword provides an explicit way to specify the default. A timing update is not needed when restoring the session.
- *compatible* Saves the session in a form that can be restored into the same or later PrimeTime releases. A timing update is always needed when restoring the session.
- *both* Saves a combined version-compatible and version-specific session that can be restored into the same or later releases. A timing update is only needed when restoring the session into a later release.

`-include` *`incl_list`*

Specifies additional data to be included by the *save_session* command. Allowed values are:

- *libraries* - Includes the loaded libraries in the saved session directory. After you restore an image that was saved with the *-include libraries* option, the current session relies on the saved image for its library data, imposing the following restrictions:
  - PrimeTime updates the *search_path* variable to include the library location within the saved image.
  - A subsequent *save_session* command cannot overwrite the saved image containing the libraries of the present session.
  - Any subsequent *save_session* command also includes the libraries.
- *gca_session* - Includes the GCA session in the saved session. This forcefully saves the GCA session. This session is used when running the PrimeTime GCA commands.
- *timing_paths* - Includes timing path collections in the saved session.
- *physical_data* - Includes ECO physical data in a version-compatible saved session. This value is valid only when the *-version compatible* option is set. (Version-specific sessions always save ECO physical data when present.)

  Note: The IC Compiler II reference libraries, specified with the *set_eco_option -physical_icc2_lib* option, are not copied to the session directory. Their original location is stored in the *session_dir*/eco_config file. If needed, edit these paths before restoring the session.

`-only_used_libraries`

By default, the *save_session* command saves references to all loaded logic libraries at the saving of the session. All of these libraries are required to restore the session later. This option saves references to only libraries that are in use, which are needed to restore the session later.

`-disable_common_data_sharing`

By default, a multi-scenario *save_session*, whether issued at the manager or at the workers via *remote_execute*, will share common data between scenarios. This option disables the feature for the present save, and the resulting scenario images contain their own (duplicate) data.

`-only_timing_paths` *`path_collection`*

Saves the given timing path collection to disk in a representation that is independent of timing data and can be subsequently restored along with any

number of other path sessions for analysis. On restore of this type of session, PrimeTime enters into the interactive multi-scenario analysis (IMSA) mode, which has a limited set of enabled commands. You cannot use this option with the *-only_used_libraries* option.

`-incremental`

Incrementally add an additional timing path collection to a saved session that was previously generated with a *save_session* command with the *-only_timing_paths* option or a previous incremental save. You cannot use this option with the *-only_used_libraries* or *-name* options; you must use the *-only_timing_paths* option.

`-name` *`session_name`*

Specifies a name for the saved timing path session, used together with the *-only_timing_paths* option.

`dir_name`

Specifies the directory to save the session to. This option is optional for the HyperScale flow. Otherwise, it is required.

If you are using a flow other than the HyperScale flow, and the named directory does not exist, the *save_session* command attempts to create it. If it already exists, the *save_session* command attempts to delete the directory before recreating it. If PrimeTime is unable to delete the directory due to a lack of write permission, an error is issued and the session is not saved.

Description

Use this command to create a repository of data to save the current PrimeTime session to. Use the *dir_name* option to specify the name of the directory to save the session. If the directory does not exist, a new one is created and the data for the session is written into the directory.

If the directory target save directory already exists, the default behavior is to always overwrite data. Otherwise, a new directory is created to write the data.

The *save_session* command does not perform an implicit *update_timing*; therefore, you can save sessions before using the *update_timing* command. Only a linked design can be saved, so the *save_session* might cause an implicit link to be performed. If there are incremental timing updates pending (the existing timing data is stale), PrimeTime performs an implicit *update_timing* as needed. If noise is to be included, an implicit *update_noise* is also performed as needed.

Note that high-capacity settings are saved and restored. Therefore, restoring a session results in the saved session's high-capacity mode overriding the mode in the session where the *restore_session* command was issued. Additionally, in multicore analysis mode, the distributed multicore analysis settings are saved in the

multicore_compute_resources.tcl file located in the *save_session* directory. During a *restore_session*, this script is sourced to launch the multicore workers.

Note: The tool does not save collections that involve timing objects, Tcl procedures, threaded multicore settings, and warning and informational message suppressions. Also, if there are other designs in PrimeTime memory, other than the linked design, they are not saved. These designs must be separately read into the restored session.

By default, the command saves a version-specific session that must be restored into the same PrimeTime version. The required version is indicated in the README file located in the session directory. Or, use the *-version compatible* option to save a version-compatible session (without timing/noise update data) that can be restored into the same or later version of PrimeTime.

An arbitrary collection of *timing_path* objects generated in a PrimeTime session can be saved using the *-only_timing_paths* option of the *save_session* command. In this case, only the *timing_path* collection and supporting data are saved, not the full design or timing information. Subsequently, multiple path sessions can be loaded into memory in a regular PrimeTime environment using the *restore_session* command.

Distributed Multi-Scenario Analysis

When *save_session* is called in a distributed multi-scenario analysis (DMSA) run, certain data, such as parasitic and physical database information, are stored in a common data directory named common_data. The images saved by each scenario contain softlinks to the data in this common data directory to avoid a duplication of storage of the data that scenarios share in common.

When *save_session* is issued directly to the manager session, the common data directory is created in the specified image directory. When *save_session* is issued to the workers via *remote_execute*, the location of the common data directory is based on whether the image directory specified to the command is relative or absolute:

- For absolute paths, the common data is located in the parent directory of the specified directory. All scenario images should share the same parent directory to maximise sharing.
- For relative paths, the common data directory is located directly under the multi-scenario working directory.

The softlink to common data in the image directory will be in the form of a relative link (a link to ../common_data) where the save session is either issued at the manager (e.g. *save_session session_name*), or if it is remotely executed at the workers and the location is specified by an absolute path (e.g. remote_execute {*save_session /absolute/path/to/session/[current_scenario]}*).

In the case where *save_session* is remotely executed at the workers and the location is specified as a relative path (e.g. *remote_execute {save_session session/*

[current_scenario]}), the softlink to common_data is an absolute path to the common_data directory directly under the DMSA working directory. To execute a multi-scenario *save_session* with common data sharing disabled, use the *-disable_common_data_sharing* option.

Version-Compatible Sessions

When the *-version compatible* option is specified, the session is stored in a format that can be restored into the same or later versions of PrimeTime.

This session format cannot store the in-memory data from a timing update; thus, a timing update is needed to perform analysis after restoring the session. This can result in slight QoR differences, particularly when restoring into a newer version of the tool.

Similarly, in-memory physical data for ECO fixing is not stored and will be reloaded when the *check_eco* command or a *fix_eco_** command is run.

Because the binary timing and ECO data is not included in the session, version-compatible session directories can be smaller than version-specific session directories.

Note: version-compatible session should be saved before any ECO optimization i.e. before fix_eco_* or any manual ECO command gets run, check user guide for flow details.

Examples

The following example saves a PrimeTime session in the state1 directory:

```
pt_shell> save_session state1
```

The following example saves a version-compatible PrimeTime session in the vc_session directory with ECO physical data:

```
pt_shell> save_session -version compatible vc_session \\
            -include {physical_data}
```

The following example generates and saves a collection of timing paths:

```
pt_shell> set paths [get_timing_paths -max 10 -nworst 10]
pt_shell> save_session -only_timing_paths $paths path_session_dir
```

See Also

- restore_session

---

save_timing_paths

Saves a collection of timing paths in a named directory so that you can later restore the collection with the *restore_timing_paths* command.

Feedback

Syntax

int *save_timing_paths*

```
path_collection
-output dir_name
```

Data Types

path_collection	collection
dir_name	string

Arguments

path_collection

Saves the given timing path collection to disk.

-output dir_name

Specifies the directory into which the timing path collection is saved.

Description

Use this command to write a collection of timing paths to a disk directory. You must specify the path collection and the directory in which to save the collection. If the directory already exists, the command overwrites it. Otherwise, the command creates a new directory.

You can later read the timing path collection into the PrimeTime tool using the *restore_timing_paths* command. The design, constraints, and version of the PrimeTime tool must be the same as when the path collection was saved.

Examples

The following example saves a timing path collection named my_paths into a directory named paths1:

```
pt_shell> set my_paths [get_timing_paths -nworst 20 -max_paths 500]
 ...
pt_shell> save_timing_paths $my_paths -output paths1
 ...
```

The following example saves a timing path collection directly into a directory named paths2:

```
pt_shell> save_timing_paths [get_timing_paths -delay_type min] -output
 paths2
 ...
```

Feedback

See Also

- get_timing_paths
- restore_timing_paths

---

save_training_data

Saves training data to a file after PrimeTime learns users' custom sizing operations.

Syntax

```
status save_training_data
   -output file_name
   [-force]
```

Data Types

```
file_name     string
```

Arguments

`-output file_name`

Writes the trianing data to the specified file.

`-force`

Forces overwriting the specified file if the file exists.

Description

The *save_training_data* command saves training data to a file after PrimeTime learns users' custom sizing operations, after the *start_learning* command enables PrimeTime to learn users' custom sizing operations. PrimeTime currently supports only ECO operations performed by one or multiple *size_cell* commands. After the *start_learning* command is executed, PrimeTime is ready to learn optimization that consists of one or multiple size_cell commands.

After PrimeTime learns custom sizing operations, the *save_training_data* command saves the training data for future usage. Later PrimeTime uses the training data to train itself with the training data and applies the learned sizing operations to new designs.

Examples

The example scripts below illustrate how to generate custom optimization training data. First perform the *start_learning* command to get PrimeTime ready for custom sizing operations. Then, apply *size_cell* commands manually or source a file that contains multiple *size_cell* commands. This is useful when a customized ECO script generates a sequence of *size_cell* commands. Once sizing operation is completed, use the *save_training_data* command to save the training data for future ECO operations.

```
pt_shell> start_learning
pt_shell> size_cell U100
pt_shell> source my_eco_changes.tcl
pt_shell> size_cell U200
pt_shell> save_training_data -out my_custom_training.td
```

The following example illustrates the *fix_eco_power* command reads the training data in the example above, trains itself with the training data, and reduces power in the current design using the training data.

```
pt_shell> set training_data_directory ./my_training_data_dir
pt_shell> fix_eco_power -training_data my_custom_training.td
```

See Also

- fix_eco_power
- start_learning
- training_data_directory

---

scale_parasitics

Scales the parasitic resistance and capacitance values in memory by specified factors.

Syntax

status *scale_parasitics*

```
[-resistance_factor r_factor]
[-ground_capacitance_factor c_factor]
[-coupling_capacitance_factor cc_factor]
[-dont_apply_to_annotated]
[net_list]
```

Data Types

```
r_factor          float
c_factor          float
cc_factor         float
net_list          list
```

Arguments

`-resistance_factor r_factor`

The scaling factor for parasitic resistance values, a positive floating-point number. For example, a factor of 1.10 increases the resistor values by 10 percent.

`-ground_capacitance_factor` *`c_factor`*

The scaling factor for net-to-ground capacitance values.

`-coupling_capacitance_factor` *`cc_factor`*

The scaling factor for cross-coupling capacitance values.

`-dont_apply_to_annotated`

Applies the specified global scaling factors only to parasitic data read in after execution of the *scale_parasitics* command, without affecting parasitic data already annotated on the design. By default, global scaling factors apply to both existing and new parasitic data. This option can be used only to specify scaling factors globally, without the *net_list* option.

`net_list`

Applies the scaling only to the specified list of nets.

Description

The *scale_parasitics* command scales parasitic resistance and capacitance values by specified factors. You can perform a single parasitic extraction run and then use this command to scale the parasitic data for variation effects, such as resistance dependence on temperature.

The command works for detailed parasitics as well as pi models. The command works irrespective of the source of the parasitic data (SPEF, GPD, DSPF, RSPF, and so on). Any subsequent delay calculations use the scaled parasitics instead of the original parasitics. If you write out parasitics using the *write_parasitics* command, the scaled parasitics are written out.

You can scale the parasitics for the whole design or just a specific list of nets. A new scaling setting overwrites any previous conflicting setting. If you scale the whole design and then a specific net, the net-specific scaling applies to the original value, not the globally scaled value.

If you read in parasitics again for previously scaled nets, the global scaling factors still apply, but net-specific scaling factors are discarded.

To report or cancel scaling factors, use the *report_scale_parasitics* or *reset_scale_parasitics* command.

Examples

The following example scales the resistance values by a factor 1.1, the ground capacitance values by a factor of 1.3, and the coupling capacitance values by a factor of 0.95. This scaling applies to the whole design and to any new parasitics read in later.

```
pt_shell> scale_parasitics -resistance_factor 1.1 \\
          -ground_capacitance_factor 1.3 \\
          -coupling_capacitance_factor 0.95
```

The following example scales the ground capacitance by a factor of 1.1 and the coupling capacitance by a factor of 0.95 for net n123. This scaling overwrites any previous global or net-specific setting for the net. If you read new parasitics for the net, the net-specific settings are discarded.

```
pt_shell> scale_parasitics -ground_capacitance_factor 1.1 \\
          -coupling_capacitance_factor 0.95 [get_nets n123]
```

See Also

- read_parasitics
- reset_scale_parasitics
- report_scale_parasitics
- write_parasitics

---

sdp

Records session diagnostic data for tool analysis and debugging by the Synopsys product support team.

Syntax

```
status sdp
   [-start]
   [-stop]
```

Data Types

None

Arguments

-start

Starts SDP session diagnostic data recording.

-stop

Stops SDP session diagnostic data recording.

Description

The Synopsys Diagnostic Platform (SDP) utility is for investigating and debugging issues with PrimeTime performance, capacity, long runtime, and crashes.

Execute the *sdp -start* command to start recording tool operating data and *sdp -stop* to stop recording data. When you exit from pt_shell, the tool writes the tool diagnostic data into a compressed tar file. You can then send the file to the Synopsys product support team for analysis and debugging.

The *sdp* command records only tool usage data. The generated tar file does not contain any customer design data, so your design remains confidential. The runtime overhead of using the command is small, typically less than 5 percent.

The *sdp* command is hidden (not displayed by the *help* command). Use the command only at the direction of the Synopsys product support team for tool analysis and debugging purposes. The generated data file can be read and analyzed only by the Synopsys product support team.

The command records the following types of data:

- OS version, CPU model, environment variables, QSC compliance
- CPU, memory, and I/O utilization
- Start/end times, running/idle/waiting status
- Stack trace at regular intervals and on a fatal crash
- CPU profiling information: time spent in each function, ...
- Hardware profiling information: cache misses, FP operations, ...
- Memory profiling information: allocated memory, memory peaks

While the *sdp* command records session data, it writes intermediate data to a directory under the current working directory. The amount of disk space used depends on the session duration, number of processes and threads, and the actions carried out during the session. The intermediate data files and directory are automatically deleted upon exit from pt_shell.

For example, a test run of 10 hours with 10 subprocesses, each having 4 to 8 threads, generated 6 GB of uncompressed data. Upon exit from pt_shell, the tool rewrote out the data in compressed format, producing a 200 MB file.

In a DMSA flow, run the *sdp -start* command at the DMSA master. The operation is queued to be performed at the start of each task executed by a worker process. Similarly, run the *sdp -stop* command at the DMSA master. The *sdp* command works in a similar manner in HyperGrid distributed analysis flows.

Examples

The following command starts SDP data recording:

```
pt_shell> sdp -start
```

Feedback

```
SDP Info: 'stack_tracer' 'crte' 'sst' 'stopwatch' started.
SDP Info: Log folder is located at
 '/remote/my_run_dir/sdp/runid_ig106_20180907_000022068'.
1
pt_shell>
```

The following command stops SDP session data recording:

```
pt_shell> sdp -stop
SDP Info: 'crte' 'sst' 'stack_tracer' 'stopwatch' stopped.
1
```

When you exit pt_shell, the tool reports the data collected by SDP:

```
pt_shell> exit

Timing updates: 4 (2 implicit, 2 explicit) (0 incremental, 4 full, 0
 logical)
Noise updates: 0 (0 implicit, 0 explicit) (0 incremental, 0 full)
Maximum memory usage for this session: 1004.42 MB
CPU usage for this session: 28 seconds
Elapsed time for this session: 2551 seconds
Diagnostics summary: 2 errors, 6 warnings, 291 informationals

Thank you for using pt_shell!

SDP Info: SDP has collected diagnostic data that is useful for analyzing
 the
runtime performance of Synopsys tools. A tar file 'sdp.tgz' is created.
A copy of the file should be sent to the Synopsys support team for
 analysis.
DO NOT MODIFY THIS FILE.

SDP Self Profile took 22 sec
```

See Also

- cputime
- mem

---

set_active_clocks

Sets a group of clocks to be active in the current analysis scope.

Syntax

status *set_active_clocks*

```
active_clock_list
[-all_clocks]
```

Feedback

Data Types

`active_clock_list` list

Arguments

`active_clock_list`

Specifies a list of clocks matching the clock names. The *active_clock_list* and *all_clocks* are mutually exclusive.

`-all_clocks`

Specifies to analyze all clocks simultaneously. The *active_clock_list* and *all_clocks* are mutually exclusive.

Description

Sets a group of clocks to be active in the current analysis scope. Provides capability to perform timing analysis for a particular set of clock domains, but not the entire chip. If a few of the clocks are set as active, the tool performs timing analysis only for these clock domains.

If this command is not specified, the tool analyzes all clocks. The last *set_active_clocks* command always overrides previous ones. If you want to analyze all clocks simultaneously again, use the *set_active_clocks* command with the *active_clock_list* option set to a value of * or use it with the *all_clocks*.

When a clock or a generated clock is created, it is active by default. If a generated clock is active itself, but its master clock is inactive, it is not analyzed. You can use the *is_active* attribute of the clock object to create a collection of active clock objects. Thus, you can freely add or remove individual clocks from this collection.

You can use the *set_active_clocks* and *set_clock_groups* commands together to simultaneously analyze multiple clocks per register without manually specifying false paths. For examples, see the *set_clock_groups* man page.

When the *set_active_clocks* command applied on crosstalk delay analysis, only the victim and aggressors driven by active clocks are considered. Therefore, the nets driven by the non-active aggressor are considered quiet. However, all nets are sensitive to static noise (as victim) even when there is no active clocks driving them. This feature could be used to set test clocks to non-active when the design is in normal or mission mode and vice versa. It is important that you set all active clocks in the design as active, as they can be aggressors to other part of the design.

This command is not supported with the *write_sdc*, *characterize_context*, and *extract_model* commands.

The *set_active_clocks* command is used to speed up update_timing for local analysis, and some global functionalities might not behave as you expect. For example, check_timing reports endpoints as unconstrained if they are captured by inactive clocks.

Examples

The following example sets only two of the existing clocks in the design to be active.

```
pt_shell> set_active_clocks {clk1 clk2}
```

The following example resets all clocks in the design to be active.

```
pt_shell> set_active_clocks [all_clocks]
```

The following example adds *clk3* to the current active clocks list.

```
pt_shell> set het [get_clock * -filter is_active==true]
pt_shell> set newfoo [add_to_collection $het [get_clocks clk3]]
pt_shell> set_active_clocks $newhet
```

The following example removes *clk3* from the current active clocks list.

```
pt_shell> set het [get_clock * -filter is_active==true]
pt_shell> set newfoo [remove_from_collection $het \\
[get_clocks * -filter "is_active==true && full_name == clk3"]]
pt_shell> set_active_clocks $newhet
```

See Also

- add_to_collection
- all_clocks
- create_clock
- create_generated_clock
- get_clocks
- set_clock_groups
- remove_clock_groups
- remove_from_collection
- report_clock

---

set_activity_derate

Sets clock activity derating factors for a specified list of instances or cell types (combinational, integrated clock gating cells).

Feedback

Syntax

```
int set_activity_derate
[-cell_type]
[-derate_factor]
derate_value
object_list
```

Data Types

cell_type	list
derate_value	float
object_list	list

Arguments

`-cell_type`

Indicates that the *derate_value* specified should be applied to combinational, or integrated clock gating cells in the clock network.

derate_value

Specifies the value of the derating factor that is applied to the toggle rate of cells in the clock network.

object_list

Specifies current design or a list of cells to which the specified clock derating factor is applied.

Description

Sets toggle rate derating factors on different cell types in the current design, or a list of cells in the current design. The clock derating factors are used as a scalar multiplicative factor in calculation of toggle rate for output net of cell.

Clock activity derating factor affects toggle rate on cells in the clock network. If clock derating factors are not specified, the value of 1.0 is assumed.

If you use the *set_activity_derate* command with the *-cell_type*, option, the specified activity derating factor is only applied to combinational or integrated clock gating cells accordingly. The derating is not applied to inverters or buffer cells in the clock network. But you can explicitly set derate on buffer/inverter using *cell_list* option.

The *object_list* option can be used to set specific derating factors on instances (cells) in the design. On each instance the *-derate_value* can be used in the same way as previously described to specify exactly how the derating factors should be applied to cell types.

If clock activity derating factors are set multiple times on the same design object or the same cell type, the later value overrides the previous value.

To set derating values back to the default (derating factor of 1.0 for every instance in the design), use the *set_activity_derate* command again with *derate_vale* value of 1.0.

Examples

The following example sets clock activity toggle rate derating factor of value 0.1 on all combinational cells in the design.

```
pwr_shell> set_activity_derate -derate_factor 0.1 -cell_type
 combinational
```

The following example sets clock activity toggle rate derating factor of value 0.1 to all cells in the design.

```
pwr_shell> set_activity_derate -derate_factor 0.1 [get_cells *]
```

The following example sets the toggle derate to 0.1 for all integrated clock gating cells in the design.

```
pwr_shell> set_activity_derate  -derate_factor 0.1 -cell_type
 clock_gating_cells
```

The following example sets a toggle derating factor of 0.1 on cell instance u1 in the design.

```
pwr_shell> set_activity_derate -derate_factor 0.1 [get_cell u1]
```

See Also

- update_power
- set_switching_activity

---

set_advanced_analysis

Turn on advanced analysis mode.

Syntax

string *set_advanced_analysis*

Arguments

None.

Description

This command turns on the advanced analysis mode. In this mode, PrimeTime will perform analysis using enhanced algorithms. In addition, some features will be enabled: the all_path_edges mode will be used in SI window overlap analysis; combined launch and capture depths will be used in AOCV analysis, replacing

separate_launch_capture_depth or separate_data_and_clock_metrics settings. Also, automatic inference of MUX cells for clock exclusivity will be enabled. These additional features can be modified or turned off after the execution of this command. Turning off the aforementioned features after executing this command will still impact the analysis. Potential degradation of runtime and capacity compared to the default mode of analysis is also expected.

It is recommended that this command be executed right before update_timing to prevent potential modification and confliction of related settings. This feature requires PrimeTime SI license.

Examples

The following example shows how to turn on advanced analysis mode:

```
pt_shell> set_advanced_analysis
Information: Checked out license 'PrimeTime-SI' (PT-019)
Changing the following variable settings:
Information: setting si_xtalk_delay_analysis_mode to all_path_edges
Information: setting timing_aocvm_analysis_mode to
 combined_launch_capture_depth
Information: setting timing_enable_auto_mux_clock_exclusivity to true
```

---

set_advanced_multi_input_switching_factor

Sets a derate of advanced multi-input switching (MIS) analysis for a specified list of library cells.

Syntax

int *set_advanced_multi_input_switching_factor*

```
-scale_factor scale factor
[-all]
-reset
[-inverse]
[object_list]
```

Data Types

scale_factor	float
object_list	list

Arguments

`-scale_factor`

Applies a scaling factor to decrease the advanced MIS derate for the specified library cell.

`-all`

Applies the command to all the library cells of the design.

`-reset`

Resets the scaling factor for the specified library cells.

`-inverse`

Applies an inverse scaling factor to increase the advanced MIS derate for the specified library cell.

`object_list`

Specifies a list of library cells for application of the scaling factor.

Description

This command defines a scaling factor that is used to scale the computed advanced MIS coefficients for the specified list of library_cells. The value of the scaling factor must be between 0.0 and 1.0. With the *-inverse* option, the scaling factor is used to increase as opposed to decrease the advanced MIS derate. In case two commands, one with the *-all* option and one with a specific library cell are specified, the library cell specification wins.

Examples

The following example changes the tool computed advanced MIS factor from 0.5 to 0.75 for the NAND2 cell.

```
pt_shell> set_advanced_multi_input_switching_factor -scale_factor 0.5
[get_lib_cell NAND2]
```

The following example changes the tool computed advanced MIS factor from 0.5 to 0.25 for the NAND2 cell.

```
pt_shell> set_advanced_multi_input_switching_factor -inverse
 -scale_factor 0.5
[get_lib_cell NAND2]
```

The following example changes the tool computed advanced MIS factor from 0.5 to 0.25 for all cells in the design.

```
pt_shell> set_advanced_multi_input_switching_factor -all -inverse
 -scale_factor 0.5
```

The following example changes the tool computed advanced MIS factor from 0.5 to 0.75 for the NAND2 library cell and from 0.5 to 0.25 for all other cells in the design.

```
pt_shell> set_advanced_multi_input_switching_factor -scale_factor 0.5
[get_lib_cell NAND2]
pt_shell> set_advanced_multi_input_switching_factor -all -inverse
 -scale_factor 0.5
```

The following example resets the scaling factor for all cells in the design.

```
pt_shell> set_advanced_multi_input_switching_factor -reset -all
```

See Also

- si_enable_multi_input_switching_analysis
- si_multi_input_switching_analysis_mode

---

set_annotated_activity_waveform

Allows user to annotate a waveform on design nets or leaf level pins.

Syntax

```
status set_annotated_activity_waveform
[-period period_value]
[-waveform edge_list]
[-value_change change_list]
object_list
```

Data Types

period_value	float
edge_list	list
change_list	list
object_list	list

Arguments

`-period` *period_value*

Defines period value of a symmetric repetitive waveform. It takes float value as input and default unit is ns (nano seconds).

`-waveform` *edge_list*

This option can only be used with -period. Defines the rise and fall transition time within specified period in order of rise transition, fall transition and so on, first edge is always input as rise transition. It can only have even number of edges. If this option is not specified, then rise edge will be assumed at start of period and a fall edge will start from the half- period.

`-value_change` *change_list*

This option is mutually exclusive with -period option. Defines list of time and signal value pair values.. Value can be 0 or 1.

object_list

object_list can be a collection of either nets or leaf level pins.Name of object should be hierarchical. This option accepts wildcard characters.

Description

This command enables user to annotate waveform on nets or leaf level pins. The command overrides annotated waveform from vector file for these pins or signals if present. Annotation type set_annotated_activity_waveform is set for nets or pins on which user has annotated waveform. Toggles specified on the leaf pins of a cell are automatically applied on the nets connected to that pin.

Examples

The following *set_annotated_activity_waveform command generates waveform on d0 net.*

```
pt_shell> set_annotated_activity_waveform -waveform { .25 .75 }
 -period .9 [get_nets d0]
pt_shell> get_switching_activity [get_net d0]
{"u_SDFFW/d0" 2 set_annotated_activity_waveform
 0 set_annotated_activity_waveform 0.571429
 set_annotated_activity_waveform}
```

The following *set_annotated_activity_waveform command generates toggles on d0 net.*

```
pt_shell> set_annotated_activity_waveform -value_change {{3, 1} {3.25,
 0} } [get_nets d0]
```

See Also

- get_switching_activity

---

set_annotated_check

Sets the setup, hold, recovery, removal, or nochange timing check value between two pins.

Syntax

string *set_annotated_check*

```
-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low |
 -width | -period
[-rise]
[-fall]
[-min]
[-max]
[-from from_pins]
[-to to_pins]
[-clock rise | fall]
```

```
[-cond sdf_expression]
[-worst]
[-increment]
[-override_increment]
[comment]
check_value
```

Data Types

```
from_pins           list
to_pins             list
sdf_expression      string
check_value         float
comment    string
```

Arguments

`-setup`

Specifies that data must be stable for the amount of time specified by the *check_value* option before the closing clock edge.

`-hold`

Specifies that data must be stable for the amount of time specified by the *check_value* option after the closing clock edge.

`-recovery`

Specifies that an asynchronous set or clear inactive edge cannot occur within the *check_value* option before the closing clock edge. This is essentially a setup requirement on an asynchronous pin, but only the inactive transition is considered.

`-removal`

Specifies that an asynchronous set or clear inactive edge cannot occur until the *check_value* option time after the closing clock edge. This is essentially a hold requirement on an asynchronous pin, but only the inactive transition is considered.

`-nochange_high`

Specifies that the data must not rise within the *check_value* option time before the clock becomes active, and must not fall until the *check_value* variable time after the clock becomes inactive. A rise data transition corresponds to the check before the activating clock edge. A fall data transition corresponds to the check after the deactivating clock edge. A rise clock transition indicates that the clock is active high. A fall clock transition indicates that the clock is active low.

`-nochange_low`

Specifies that the data must not fall within the *check_value* option time before the clock becomes active, and must not rise until the *check_value* option time after the clock becomes inactive. A fall data transition corresponds to the check before the activating clock edge. A rise data transition corresponds to the check after the deactivating clock edge. A rise clock transition indicates that the clock is active high. A fall clock transition indicates that the clock is active low.

`-width`

Specifies the minimum pulse width constraint for a clock pin. The timing check is defined on the *-from* pin, so the related pin (*-to*) should either be the same or it can be omitted.

`-period`

Specifies the minimum period constraint for a clock pin. The timing check is defined on the *-from* pin, so the related pin (*-to*) should either be the same or it can be omitted.

`-rise`

Specifies that the timing check is for the data rise transition. If you do not specify either the *-rise* or *-fall* options, both values are set.

`-fall`

Specifies that the timing check is for the data fall transition. If you do not specify either the *-rise* or *-fall* options, both values are set.

`-min`

Use this option only if the design is in min_max mode (min and max operating conditions). Specifies the minimum timing check for both data rise and data fall transitions.

`-max`

Use this option only if the design is in min_max mode (min and max operating conditions). Specifies the maximum timing check for both data rise and data fall transitions.

`-from` *`from_pins`*

Specifies a list of leaf cell clock pins that are the startpoints of the timing arcs for which checks are annotated. There must be a corresponding timing arc between the *from_pins* and the *to_pins* options.

`-to` *`to_pins`*

Specifies a list of leaf cell data pins that are the endpoints of the timing arcs for which checks are annotated. There must be a corresponding timing arc between the *from_pins* and the *to_pins* options.

`-clock rise | fall`

Specifies whether the check is for clock rising or falling. By default, checks for both clock rise and fall are set.

`-cond` *`sdf_expression`*

Use this option only if the library has a condition attached to the specified timing check arc; otherwise, an error message is generated. Specifies the condition for which the annotated check is valid. The syntax of the condition must match the condition specified in the library using the sdf_cond construct. The syntax is the same one used in the Standard Delay Format (SDF).

`-worst`

This option is not currently implemented, so it is ignored.

`-increment`

Specifies that the delay value is to be added to the calculated check value of the specified timing arc. If an annotated incremental value already exist, then the new value will be added on the existing one.

`-override_increment`

Overwrite incremental values instead of adding them on the already existing. Applies only for annotating incremental values.

`comment`

Associates a string description with the command for tracking. The comment string must begin and end with double quotation marks ("). Currently, this option is disabled by default. To enable it set pt_enabled_comment_option gvar to true.

`check_value`

Specifies the timing check value between pins on the same cell, in units consistent with the logic library used during optimization. For example, if the logic library specifies delay values in nanoseconds, the *check_value* option must be expressed in nanoseconds. The *check_value* option can be negative.

Description

Annotates a timing check arc between two or more pins on a cell or a net in the current design. This can be appropriate after place and route, for technologies where the timing check value varies for different instances, and the library timing check values do not provide sufficient accuracy.

If the design is not linked, it is linked automatically by the *set_annotated_check* command.

You can use the *set_annotated_check* command for pins at lower levels of the design hierarchy. You can specify pins in the format of *INSTANCE1/INSTANCE2/PIN_NAME*.

To list annotated timing check values, use the *report_annotated_check* command. To remove annotated timing check values from a design, use the *remove_annotated_check* or *reset_design* commands. To see the effect of the *set_annotated_check* command for a specific instance, use the *report_timing* command.

Examples

The following example annotates a setup time of 2.1 units between the CP clock pin of cell instance u1/ff12 and the D data pin of the same cell instance.

```
pt_shell> set_annotated_check -setup 2.1 -from u1/ff12/CP -to u1/ff12/D
```

The following example annotates a check for the maximum value of 3.0 units for a minimum pulse width of a clock signal in 'high' state at the u1/ff12/CP clock pin. The annotation is then reported by *report_annotated_check*:

```
pt_shell> set_annotated_check -width 3.0 -clock rise -max  -from
 u1/ff12/CP
pt_shell> report_annotated_check -width -list_annotated
****************************************
Report : annotated_check
         -width
         -list_annotated
 ...
****************************************

Backannotated cell min_pulse_width check arcs:
-----------------------------------------------
1. u1/ff12/CP -> u1/ff12/CP  (r:0.10/3.00  f:NA  sense:
 clock_pulse_width_high)

                              |           |           |    NOT    |
    Timing check type         |    Total  | Annotated | Annotated |
------------------------------+-----------+-----------+-----------+
cell min pulse width arcs     |       20  |        1  |       19  |
```

The following examples illustrates how the incremental values are annotated and how incremental values are used.

On the example below a value of 1.0 unit is annotated on the setup time between the clock pin and the data pin of a cell instance. After that an incremental setup value of 0.5 units is annotated on the same arc. The final value of the setup constraint on this arcs is 1.5.

```
pt_shell> set_annotated_check -setup -from ff/CP -to ff/D 1.0
pt_shell> set_annotated_check -increment -setup -from ff/CP -to ff/D 0.5
```

Feedback

Continuing the above example, the following command annotates another 0.6 unit on this setup arc. The new setup value for this arc is now 2.1, where 1.0 is the fully annotated value and 1.1 (0.5 + 0.6) is the incremental annotated value.

```
pt_shell> set_annotated_check -increment -setup -from ff/CP -to ff/D 0.6
```

The overriding of the previously annotated incremental values on a timing arc can be achieved with option *-override_increment*. The impact that the following command has on the above example is to override the current increment value of 1.1 with a new one of 0.7. The new setup value is now 1.7 since 1.0 was the initial fully annotated value and 0.7 in the new increment value.

```
pt_shell> set_annotated_check -increment -override_increment -setup -from
 ff/CP -to ff/D 0.7
```

The incremental annotated values equally interacts with fully annotated values and calculated values. They can also be removed with the usage of the *remove_annotated_check* or *reset_design* commands.

See Also

- current_design
- link
- read_sdf
- remove_annotated_check
- report_annotated_check
- report_timing
- reset_design

---

set_annotated_clock_network_power

Annotate power on clock networks.

Syntax

string *set_annotated_clock_network_power*

```
[-internal_power internal_power]
[-switching_power switching_power]
[-leakage_power leak_power]
[-total_power total_power]
[-clock domain_clock]
```

Feedback

Data Types

internal_power	float
switching_power	float
leak_power	float
total_power	float
domain_clock	string

Arguments

`-internal_power` *internal_power*

Specifies the annotated internal power value on the clock network.

`-switching_power` *switching_power*

Specifies the annotated switching power value on the clock network.

`-leakage_power` *leak_power*

Specifies the annotated leakage power value on the clock network.

`-total_power` *total_power*

Specifies the annotated total power value on the clock network. It is treated as internal power in the reports.

`-clock` *domain_clock*

Specifies the clock in the related clock network on which the power values are annotated.

Description

The *set_annotated_clock_network_power* command provide the capability to annotate the power values on the clock networks in the reports from PrimePower. If this command is used before the *report_power* command, PrimePower uses the annotated clock network power values in the summary report. There is also a separate annotated clock network power report embedded in the regular summary power report. The *clock_network* group power in the summary report excludes the clock network objects whose power values are annotated. An attribute 'e' marks such a situation in the report.

The power values in the unit of Watt are specified by either *-total_power* option of a combination of *-internal_power*, *-switching_power* and *-leakage_power* options, If *-total_power* option is used, the power values are also treated as internal power in the reports to make the results consistent. If the options of *-switching_power*, *-internal_power* and/or *-leakage_power* are used, the values are annotated on internal, switching and/or leakage power respectively. The total power is the sum of these three components. These three options can be used together or separately. But they cannot be used together with the *-total_power* option; otherwise, the CMD-001 error message is issued and the command fails.

By default, the power values aree annotated on the whole clock network in the design. If *-clock* option is used, the power values are annotated on the collection of clock network objects of the corresponding clock domain. Only one clock is allowed for one command. Multiple *set_annotated_clock_network_power* commands can be used to specify annotated power of multiple clock domains separately. The annotated clock network power reports list them separately.

If two *set_annotated_clock_network_power* commands are specified for the same clock domain (or the whole clock network) and the power types (internal, switching, leakage or total) are the same, The power values of the later command overrides the earlier one. If they are not the same power type, both power values are kept and reported.

The *set_annotated_clock_network_power* command can work with *report_power -groups group_list* only when the group_list contains the default 'clock_network' power group; otherwise, an PWR-296 error message is issued and the *report_power* command fails.

The *set_annotated_clock_network_power* command causes the cells in the related clock network not to be reported by *report_power -cell* command. The other options in the *report_power* command are not affected.

The annotated clock network power has higher priority than the estimated clock network power. If both features are invoked in the *report_power* command, a PWR-295 warning message is issued and annotated clock network power values are used in power reports.

To remove the annotated clock network power values already specified for the design, use the *remove_annotated_clock_network_power* command. The *report_power* command reverts back to its original behavior.

Examples

In the following example, the internal, switching power are annotated separately for clock domain CKA and CKB.

```
pt_shell> create_clock -name CKA -period 10 clka
pt_shell> create_clock -name CKB -period 5 clka
pt_shell> set_annotated_clock_network_power -internal 1.0e-03 -switching
 2.0e-03 -clock CKA
pt_shell> set_annotated_clock_network_power -switching 2.0e-03 -clock CKB
pt_shell> report_power
```

The following is the output from the above commands

```
****************************************
Report : Statistical Average Power
Design : my_design
Version: my_version
Date   : ....
****************************************

  Attributes
```

Feedback

```
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group
      e  -  Annotated clock network power excluded

                        Internal  Switching  Leakage    Total
Power Group             Power     Power      Power      Power    (%)
 Attrs
------------------------------------------------------------------------
------
io_pad                     0.0000    0.0000    0.0000    0.0000 (0.00%)
memory                     0.0000    0.0000    0.0000    0.0000 (0.00%)
black_box                  0.0000    0.0000    0.0000    0.0000 (0.00%)
clock_network           3.536e-03    0.0000    0.0000 3.536e-03 (36.81%)
 ei
register                3.060e-03    0.0000 1.020e-05 3.071e-03 (31.96%)
combinational              0.0000    0.0000    0.0000    0.0000 (0.00%)
sequential                 0.0000    0.0000    0.0000    0.0000 (0.00%)


                        Internal  Switching  Leakage    Total
Clock                   Power     Power      Power      Power    (%)
 Attrs
------------------------------------------------------------------------
------
CKB                     0.0000     2.000e-03  0.0000    2.000e-03
CKA                     1.000e-03  0.0000     0.0000    1.000e-03
------------------------------------------------------------------------
------
All Annotated Clocks    1.000e-03  2.000e-03  0.0000    3.000e-03
 (31.23%)


  Net Switching Power  = 2.000e-03   (20.82%)
  Cell Internal Power  = 7.596e-03   (79.07%)
  Cell Leakage Power   = 1.020e-05   ( 0.11%)
                         ---------
Total Power            = 9.607e-03  (100.00%)
```

See Also

- remove_annotated_clock_network_power
- report_power

---

set_annotated_delay

Sets an annotated net or cell delay value between two pins, or modifies an existing net or cell delay.

Feedback

Syntax

string *set_annotated_delay*

```
-cell | -net
[-rise]
[-fall]
[-min]
[-max]
[-load_delay net | cell]
[-from from_pins]
[-to to_pins]
[-cond sdf_expression]
[-increment]
[-delta_only]
[-worst]
[-of_objects objects]
delay_value
```

Data Types

from_pins	list
to_pins	list
sdf_expression	string
objects	list
delay_value	float

Arguments

`-cell`

Annotates a delay on a cell, from an input to an output of the cell.

`-net`

Annotates a delay on a net, from the output of a cell to the input of another cell.

`-rise`

Applies the delay only to rising transitions.

`-fall`

Applies the delay only to falling transitions.

By default, the delay applies to both rising and falling transitions.

`-min`

Applies the delay only to the minimum operating condition in min-max mode.

`-max`

Applies the delay only to the maximum operating condition in min-max mode.

The min-max mode is enabled by the *set_operating_conditions* command with the *-min* and *-max* options.

`-load_delay net | cell`

Specifies whether load delays are included in *net* delays or *cell* delays. The default is *cell* delays.

Load delay is the portion of cell delay resulting from the capacitive load of the net the cell is driving.

All timing arcs of the same net or of the same cell must be annotated with the same type of load delay setting.

`-from` *`from_pins`*

Specifies a list of leaf-cell pins and top-level ports that are the startpoints of the timing arcs for which delays are annotated.

`-to` *`to_pins`*

Specifies a list of leaf-cell pins or top-level ports that are the endpoints of the timing arcs on which delays are annotated.

You must use the *-from* or *-to* option, or both, to specify the startpoints and endpoints; or the *-of_objects* option to specify a collection of timing arcs on which to annotate the specified delay.

`-cond` *`sdf_expression`*

Specifies the logical condition for which the annotated delay is valid. The syntax of the expression must match the condition specified in the library using the Liberty construct sdf_cond. Use this option only if the library has a condition attached to the specified delay timing arc.

`-increment`

Adds the specified delay value to the existing delay of the timing arc, instead of replacing it.

If the existing timing arc is conditional, you must also use the *-cond* option to specify the condition. Otherwise, the command has no effect. To apply an incremental change to multiple conditional timing arcs, use multiple *set_annotated_delay* commands with different *-cond* option settings.

`-delta_only`

Adds the specified delay value to the existing net delay value calculated by the tool, instead of replacing it. Use this option only with the *-net* option.

`-worst`

This option is accepted as valid syntax but is not used by the PrimeTime tool, so it is ignored.

`-of_objects` *objects*

Specifies a collection of timing arcs created with the *get_timing_arcs* command on which to annotate the delay value. You cannot use this option at the same time as the *-from* or *-to* option.

delay_value

Specifies a new annotated delay value, or an incremental change in the delay value if the *-increment* or *-delta_only* option is used. Specify the value in time units consistent with the logic library.

Description

This command annotates delay values in the design by specifying either a new delay value or an incremental change to an existing delay. You must use the *-cell* or *-net* option to annotate either a cell delay or a net delay.

You must use either the *-from* and *-to* options to specify the startpoints and endpoints of the timing arcs to be annotated or the *-of_objects* option to specify a collection of timing arcs on which to annotate the specified delay. You create a collection of timing arcs using the *get_timing_arcs* command.

Cell delay is the delay from an input pin to an output pin of a leaf-level cell. These delays are defined in the library description of the cell and typically vary with the slew of the input signal and the size of the output load of the cell. Some cell delays are conditional, depending on the logic states of some of the cell inputs.

Net delay is the delay through a net, from an output pin of a leaf-level cell or a top-level input port to an input pin of another leaf-level cell or a top-level output port. The PrimeTime tool calculates net delays from the driver and load characteristics and the RC properties of the net.

To annotate load-dependent delays due to driving cells on input ports, use the *-cell* option, omit the *-from* option, and specify the input ports with the *-to* option.

To see the timing effects of the *set_annotated_delay* command, run the *report_timing* command. In the generated timing reports, annotated delay values are marked with an asterisk (*) character.

To report annotated delay values and locations, use the *report_annotated_delay* command with the *-list_annotated* option.

Load delay, also known as extra source gate delay, is the portion of cell delay caused by the capacitive load of the driven net. Some delay calculators consider load delay part of the cell delay and others consider it part of the net delay.

If your annotated delay values (for either cell or net) assume that load delay is part of the cell delay, use the *-load_delay cell* option. If your delay values assume that load delay is included in the net delay, use the *-load_delay net* option. By default, load delays are assumed to be included in cell delays, so they appear in the *report_timing* path listings.

By default, the specified delay value overrides the calculated cell or net delay value. To add the delay value to the existing delay (instead of overriding), use the *-increment* option. If the timing arc is conditional, you must also use the *-cond* to specify the matching logical condition of the arc.

To add a delay value to a net, use the *-net* option with the *-delta_only* option. After that, if you annotate a net delay again on the same net without the *-delta_only* option, the earlier delta delay setting is ignored, based on the assumption that the latest annotation includes any desired delta delay. If you later remove the annotated full net delay from the net, the delta delay setting again takes effect.

You can use the *set_annotated_delay* command for pins at lower levels of the design hierarchy. Specify these pins using the format INSTANCE1/INSTANCE2/PIN_NAME.

To remove the annotated cell or net delay values from a design, use the *remove_annotated_delay* or *reset_design* command.

Examples

The following example annotates a cell delay of 20 time units between input pin A and output pin Z of cell instance U1/U2/U3. The specified delay value includes the load delay.

```
pt_shell> set_annotated_delay -cell -load_delay cell 20 \\
          -from U1/U2/U3/A -to U1/U2/U3/Z
```

The following example annotates a rise net delay of 1.4 time units between output pin U1/Z and input pin U2/A. The specified delay value for this net does not include load delay.

```
pt_shell> set_annotated_delay -net -rise 1.4 -load_delay cell \\
          -from U1/Z -to U2/A
```

The following example annotates a rise net delay of 12.3 time units between the same output pins. In this example, the net delay value includes load delay.

```
pt_shell> set_annotated_delay -net -rise 12.3 -load_delay net \\
          -from U1/Z -to U2/A
```

The following script creates a collection containing a timing arc and then sets the annotated delay for that timing arc.

```
set my_arc [get_timing_arcs -from I_TOP/I_CORE/U120/Z -to
 I_TOP/I_CORE/reg15/D]
set_annotated_delay -net -of_objects $my_arc 0.123
```

Feedback

See Also

- current_design
- link
- read_sdf
- remove_annotated_delay
- report_annotated_delay
- report_timing
- reset_design
- set_operating_conditions

---

set_annotated_power

Annotates power on unresolved black box cells or leaf cells.

Syntax

status *set_annotated_power*

```
[-internal_power internal_value]
[-leakage_power leakage_value]
[-switching_power switching_power_value]
[-rails rail_list]
[-pg_pin pg_pin_list]
[-clock clock_list]
object_list
```

Data Types

internal_value	float
leakage_value	float
switching_power_value	float
rail_list	list
pg_pin_list	list
object_list	list
clock_list	list

Arguments

`-internal_power` *internal_value*

Specifies the internal power (in watts) to be annotated.

`-leakage_power` *leakage_value*

Specifies the leakage power (in watts) to be annotated.

`-switching_power` *switching_power_value*

Specifies the switching power (in watts) to be annotated.

`-rails` *rail_list*

Specifies a list of rails/power supply nets on which power is annotated. This option applies the annotated power values to each power supply net (or rail in non-UPF mode) in the list. This feature is valid only if the *power_enable_multi_rail_analysis* variable is set to *true*.

`-pg_pin` *pg_pin_list*

Specifies a list of pg_pins on which power is annotated. This option applies the annotated power values to each pg_pin in the list. This feature is used in both UPF and non-UPF mode. It is valid only if *power_enable_multi_rail_analysis* variable is set to *true*.

`-clock`

Specifies a clock on which power is annotated. This option applies the annotated power values on clock. This feature is valid only if cell is a hierarchical cell.

object_list

Specifies a list of cells on which to annotate internal_power or leakage_power, or a list of nets on which annotate switching_power.

Description

The *set_annotated_power* command annotates the internal and leakage power on the specified cells and annotates the switching power on the specified nets. The command should be used mostly on unresolved black box cells, where internal and leakage power cannot be estimated. However, the command also works for any leaf cells and hierachical cells, where the annotated internal and leakage power overrides the internally estimated internal and leakage power. The internally estimated power, if available, is overridden by the annotated power number, but is resurrected if the annotated power is removed.

Switching power (power of the nets connected to the output pins of the cell) can be annotated using list of nets.

The annotated power is average power, so it has no effect on power waveforms.

Examples

The following example shows how power is annotated, removed and reported.

```
pt_shell> set_annotated_power -internal_power 0.1 -leakage_power 0.2
 U0/U1
1
pt_shell> set_annotated_power -internal_power 0.3 -leakage_power 0.4 U0
```

Feedback

```
1
pt_shell> set_annotated_power -switching_power .25 out
1
pt_shell> set_annotated_power -switching_power .25  n1

****************************************
Report : annotated_power
        -list_annotated
****************************************

Annotated cell powers:
------------------------------
1.  U0  (internal: 0.3  leakage: 0.4)
2.  U0/U1  (internal: 0.1  leakage: 0.2)

Annotated net powers:
------------------------------
1.  out                    (switching : 0.25       )
2.  n1                     (switching : 0.25       )


                             |           |           |    NOT    |
     Cell type               |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
unresolved black-box cell    |         0 |         0 |         0 |
leaf cell                    |         5 |         1 |         4 |
hierarchical cell            |         1 |         1 |         0 |
net                          |        11 |         2 |         9 |
-----------------------------+-----------+-----------+-----------+
                             |        17 |         4 |        13 |

1
pt_shell> remove_annotated_power n1
1
pt_shell> report_annotated_power -list_annotated
****************************************
Report : annotated_power
****************************************
Annotated cell powers:
------------------------------
1.  U0  (internal: 0.3  leakage: 0.4)
2.  U0/U1  (internal: 0.1  leakage: 0.2)

Annotated net powers:
------------------------------
1.  out                    (switching : 0.25       )

                             |           |           |    NOT    |
     Cell type               |    Total  | Annotated | Annotated |
-----------------------------+-----------+-----------+-----------+
unresolved black-box cell    |         0 |         0 |         0 |
leaf cell                    |         5 |         1 |         4 |
```

```
hierarchical cell             |         1  |         1  |         0  |
net                           |        11  |         1  |        10  |
------------------------------+------------+------------+------------+
                              |        17  |         3  |        14  |
1
```

See Also

- remove_annotated_power
- report_annotated_power

---

set_annotated_transition

Sets the transition time annotated on specified pins in the current design.

Syntax

int *set_annotated_transition*

```
[-rise]
[-fall]
[-min]
[-max]
[-sms_scenarios sms_scenarios_list]
[-delta_only]
slew_value
pin_list
```

Data Types

```
pin_list            list
slew_value          float
sms_scenarios_list  collection
```

Arguments

`-rise`

Indicates that the *slew_value* variable represents the data rise transition time.

`-fall`

Indicates that the *slew_value* variable represents the data fall transition time.

`-min`

Indicates that the *slew_value* variable represents the minimum transition time. Use this option only if the design is in min-max mode (min and max operating conditions).

`-max`

Indicates that the *slew_value* variable represents the maximum transition time. Use this option only if the design is in min-max mode (min and max operating conditions).

`-delta_only`

Indicates that the *slew_value* variable represents a delta transition time that is added to the transition time computed by delay calculation.

`slew_value`

Specifies the transition time of the specified pins or ports, in units consistent with the technology library used during optimization. For example, if the technology library specifies delay values in nanoseconds, the *slew_value* variable must be expressed in nanoseconds. If used with the *-delta_only* option, the *slew_value* variable can be a negative number.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this transition is applied. When this option is not given the transition applies to all SMS scenarios. This collection is created by *get_sms_scenarios*.

`pin_list`

Specifies a list of pins or ports that is annotated with the transition time *slew_value*.

Description

The *set_annotated_transition* command specifies an annotated transition time value on pins or ports in the design. This transition time value overrides the regular tool-computed transition time value computed at (and propagated from) those objects.

The *set_annotated_transition* command can be used for pins at lower levels of the design hierarchy. Pins are specified as "INSTANCE1/INSTANCE2/PIN_NAME."

To remove annotated transition times, use the *remove_annotated_transition* command.

Examples

The following example annotates a rising transition time of 0.5 units and a falling transition time of 0.7 units on input pin *A* of cell "U1/U2/U3".

```
pt_shell> set_annotated_transition -rise 0.5 [get_pins U1/U2/U3/A]
pt_shell> set_annotated_transition -fall 0.7 [get_pins U1/U2/U3/A]
```

Feedback

See Also

- remove_annotated_transition
- report_timing
- reset_design
- set_annotated_delay

---

set_aocvm_coefficient

Specifies the advanced on-chip variation (AOCV) coefficients on library cells, library timing arcs, and cells.

Syntax

status *set_aocvm_coefficient*

```
coefficient
object_list
```

Data Types

```
coefficient     float
object_list     list
```

Arguments

coefficient

Specifies a coefficient value, which should be a floating-point value greater than zero.

object_list

Specifies a list of library cells, library timing arcs, or leaf cells on which the coefficient is set.

Description

Sets AOCV coefficients on library cells, library timing arcs, and leaf cells. Coefficients are used to calculate cell path depths; the default depth increment for a cell arc in a path is 1, but you can specify another coefficient value with this command. AOCV coefficients are not required for an AOCV analysis; however, their use might increase the accuracy of the analysis.

The tool applies the coefficients in the following order of priority for cell arcs (in decreasing order of precedence):

```
1) Cell
2) Library timing arc
```

Feedback

```
3) Library cell
4) 1.0 (default)
```

To display AOCV coefficients, use the *report_aocvm -coefficient* command. To remove AOCV coefficients from a cell, use the *remove_aocvm -coefficient* or the *reset_design* command.

Examples

The following examples specify AOCV coefficients on library cells and library timing arcs.

```
pt_shell> set_aocvm_coefficient 1.2 [get_lib_cells lib/AN2]
pt_shell> set_aocvm_coefficient 2.5 [get_lib_timing_arcs -from lib/AN2/A
 -to lib/AN2/Z]
```

See Also

- read_aocvm
- remove_aocvm
- report_aocvm
- timing_aocvm_analysis_mode

---

set_aocvm_table_group

Associates an AOCV named table group or Liberty-based AOCV derating group with the list of specified design objects.

Syntax

status *set_aocvm_table_group*

```
aocvm_table_group_name
[object_list]
```

Data Types

```
aocvm_table_group_name     string
object_list                list
```

Arguments

aocvm_table_group_name

Specifies the name of the AOCV table set.

object_list

Specifies the list of design objects.

Feedback

Description

The *set_aocvm_table_group* command can operate on either hierarchical cells in the top-level design or on lib_cells.

When operating on hierarchical cells in the top-level design, it sets the association between a named AOCV table group and a hierarchical instance. An AOCV table set is a collection of AOCV tables with the same name. Names are specified using the group_name field in the AOCV table syntax.

When associations are specified between hierarchical cells in the design and named AOCV table sets, a cell (or net) derives its AOCV derating from the table set associated with the hierarchical cell or design that (a) fully contains the cell or net, and (b) is the lowest level (closest ancestor) hierarchical cell or design with an associated AOCV table set. If no such hierarchical cell or design containing the cell or net is found, the unnamed AOCV table set is used.

Note: A net inherits the AOCV derating set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV table set), the lowest level hierarchical cell that fully contains the net is not the block. Therefore, the net derating does not come from an AOCV table set that contains the block and fully encloses the net.

When operating on library_cells the *set_aocvm_table_group* command can be used to assign a Liberty-based AOCV table from a Liberty library to the given lib cell. The Liberty AOCV format allows for tables, which are not assigned to any library cell by default, to still be referenced by name. This command provides the ability to setup this reference. The syntax for the P/AOCVM derate group to be referenced follows this pattern:

lib_name/[*lib_cell_name*]/*derate_group_name*

The lib_cell_name is optional and applicable only for lib_cell based table. A lib_cell based can only be assigned to a lib cell of the same name.

Examples

The following example associates the hierarchical cell H1 with the AOCV table set A1.

```
pt_shell> set_aocvm_table_group A1 [get_cells H1]

pt_shell> sh cat test.aocvm

version:        3.0
group_name:     A1
object_type:    lib_cell
rf_type:        rise
delay_type:     cell
derate_type:    late
object_spec:    my_lib/*
voltage:        1.02
```

```
depth:          1 2 3
distance:       100 1000
table:          1.20 1.10 1.08
                1.22 1.15 1.11
```

See Also

- get_timing_paths
- read_aocvm
- remove_aocvm
- report_aocvm
- report_timing
- reset_aocvm_table_group
- set_aocvm_coefficient
- write_binary_aocvm

---

set_app_var

Sets the value of an application variable.

Syntax

string *set_app_var*

```
-default
var
value
```

Data Types

```
var         string
value       string
```

Arguments

`-default`

Resets the variable to its default value.

`var`

Specifies the application variable to set.

`value`

Specifies the value to which the variable is to be set.

Feedback

Description

The *set_app_var* command sets the specified application variable. This command sets the variable to its default value or to a new value you specify.

This command returns the new value of the variable if setting the variable was successful. If the application variable could not be set, then an error is returned indicating the reason for the failure.

Reasons for failure include:

- The specified variable name is not an application variable, unless the application variable *sh_allow_tcl_with_set_app_var* is set to true See the *sh_allow_tcl_with_set_app_var* man page for details.
- The specified application variable is read only.
- The value specified is not a legal value for this application variable

Examples

The following example attempts to set a read-only application variable:

```
prompt> set_app_var synopsys_root /tmp
Error: can't set "synopsys_root": variable is read-only
       Use error_info for more info. (CMD-013)
```

In this example, the application variable name is entered incorrectly, which generates an error message:

```
prompt> set_app_var sh_enabel_page_mode 1
Error: "sh_enabel_page_mode" is not an application variable
        Use error_info for more info. (CMD-013)
```

This example shows the variable name entered correctly:

```
prompt> set_app_var sh_enable_page_mode 1
1
```

This example resets the variable to its default value:

```
prompt> set_app_var sh_enable_page_mode -default
0
```

See Also

- get_app_var
- report_app_var
- write_app_var

Feedback

set_case_analysis

Specifies a constant logic value or transition type for a list of ports or pins.

Syntax

```
status set_case_analysis
  value
  port_or_pin_list
  comment
```

Data Types

value	string
port_or_pin_list	list
comment	string

Arguments

value

Specifies a constant logic value or a transition type assigned to the given ports or pins. These are the valid values:

- *0* or *zero* - Constant logic 0
- *1* or *one* - Constant logic 1
- *static* - Constant logic 0 or logic 1, no transitions
- *rise* or *rising* - Only rising transitions are considered
- *fall* or *falling* - Only falling transitions are considered

port_or_pin_list

Specifies the ports or pins affected by the case analysis setting.

-comment *comment_string*

Associates a string description with the command for tracking purposes. When writing out SDC, this option is useful for tagging the methodology or tool that originally synthesized the command. You must begin and end the *comment_string* with double quotation marks ("). Currently, this option is disabled as a default. To enable it set *pt_enabled_comment_option* gvar to TRUE.

Description

Case analysis specifies a list of ports or pins that have a constant logic value (*0*, *1*, or *static*) or only one type of transition (*rising* or *falling*). You can use case analysis settings to place the design into a given operating mode without altering the netlist.

When case analysis is specified as a constant value (*0*, *1*, or *static*), this value is propagated through the network where the constant value controls the traversed logic. For example, if you specify a constant value of 0 at an input of a NAND gate, the output of the NAND gate has a constant value of 1. This value is then propagated further to all cells driven by this output.

A net with the *static* logic constant value can be either 0 or 1 but cannot change, and therefore cannot act as an aggressor in crosstalk analysis. If one input of an 2-input AND gate is *static* and the other has a constant logic value of *1*, the output is *static*; or if the other input has a constant logic value of *0*, then the output is a constant logic *0*.

In case of conflicting values, the following rules apply:

- A *set_case_anlaysis* setting has priority over a built-in constant value such as a logic constant in the Verilog netlist.
- A newer *set_case_anlaysis* setting has priority over an older one set on the same port or pin.
- A *set_case_anlaysis* value set directly on a port or pin has priority over a conflicting case analysis value propagated to that port or pin.
- Where propagated case analysis values are in conflict, logic *0* has the highest priority, then the *static* case setting, then logic *1*.

In the case of a leaf pin, the constant is propagated forward only. No backward constant propagation is performed unless a hierarchical driver exists with a conflicting case value.

By default, when a logic constant value is propagated to a net, design rule checking is disabled for that net (for example, *set_max_fanout* and *set_max_capacitance* constraints). To enforce checking of the maximum capacitance rule for these nets, even with case analysis, set the *timing_enable_max_capacitance_set_case_analysis* variable to *true*.

If the case analysis value is either *rising* or *falling*, timing analysis considers only that transition type for the net. The case analysis information is used by all analysis commands.

You can use both case analysis (*set_case_analysis*) and mode analysis (*set_mode*) to fully specify the operating mode of a design. For example, a design might contain at test mode called TESTMODE, enabled by control input. You can specify the TESTMODE operating mode with the *set_mode* command, and also set the control input to a constant value to disable the test circuitry using the *set_case_analysis* command.

In the PrimePower tool, you can use the *set_case_analysis* command for power analysis in both averaged mode and time-based mode. Values set by the *set_case_analysis* command are also used for internal logic simulation to get the switching activity for nonannotated nets. However, the *-rising* and *-falling* options are ignored during power calculation. Simulation results from VCD or SAIF have higher priority than *set_case_analysis* settings.

Feedback

Examples

The following command sets the IN1 port to constant logic 0.

```
pt_shell> set_case_analysis 0 IN1
```

The following command sets the IN1 port to static, which means that it can be either constant logic 0 or constant logic 1. For the connected net, no transitions can occur and DRC rule checking is skipped.

```
pt_shell> set_case_analysis static IN1
```

The following command sets the rising transition case on pins U1/U2/A and U1/U3/CI. Timing analysis considers only rising (not falling) transitions on these pins.

```
pt_shell> set_case_analysis rising {U1/U2/A U1/U3/CI}
```

The following example disables the design mode called TESTMODE and sets the TEST_PORT port to a constant logic value *0*.

```
pt_shell> remove_mode TESTMODE U1/U2
pt_shell> set_case_analysis 0 TEST_PORT
```

See Also

- remove_case_analysis
- report_case_analysis
- report_timing
- set_case_sequential_propagation
- set_mode
- case_analysis_log_file
- case_analysis_propagate_through_icg
- case_analysis_sequential_propagation
- disable_case_analysis
- disable_case_analysis_ti_hi_lo
- timing_enable_max_capacitance_set_case_analysis
- timing_enable_max_transition_set_case_analysis

Feedback

set_case_sequential_propagation

Specifies sequential cell instances or library cells to be enabled for sequential propagation.

Syntax

```
status set_case_sequential_propagation
  cellinstance_or_libcell_list
```

Data Types

```
cellinstance_or_libcell_list     list
```

Arguments

cellinstance_or_libcell_list

Lists sequential cell instances or library cells to be enabled for sequential propagation.

Description

Selective sequential propagation allows you to specify which cell instances are enabled for sequential propagation. To enable selective sequential propagation, the variable *case_analysis_sequential_propagation* must be set to *never*.

You can specify cell instances or library cells in the same command. If a library cell is specified, all cell instances of the library cell are enabled for sequential propagation. Cell instances and library cells are tracked separately. Therefore, if a cell instance and its library cell have been enabled for sequential propagation, and later you intend to completely disable the cell for sequential propagation, then both its cell instance and library cell must be removed (see *remove_case_sequential_propagation*).

The output of the *write_script* command includes *set_case_sequential_propagation*. However, the output of the *write_sdc* command does not include *set_case_sequential_propagation* because it is not a Synopsys Design Constraints (SDC) command.

Examples

The following example enables one cell instance for sequential propagation. FF1 is the instance name of the sequential cell.

```
pt_shell> set_case_sequential_propagation FF1
pt_shell> report_case_analysis -sequential_propagation
****************************************
Report : case_analysis
        -sequential_propagation
 ...
****************************************
```

```
------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 1
------------------------------------------------------------
Cell Instance : 'FF1'
------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 0
------------------------------------------------------------
```

The following example enables one library cell for sequential propagation. If the library cell is specified, all cell instances in the design with the same library cells are enabled for sequential propagation. To specify a library cell to be enabled for propagation, provide the *library name/library cell* format as shown in the following example.

```
pt_shell> set_case_sequential_propagation snps_test/D_FF
pt_shell> report_case_analysis -sequential_propagation
**************************************
Report : case_analysis
        -sequential_propagation
 ...
**************************************

------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 0
------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 1
------------------------------------------------------------
Lib Cell : 'snps_test/D_FF'
```

The following example enables both a cell instance and a library cell for sequential propagation.

```
pt_shell> set_case_sequential_propagation { FF1 snps_test/D_FF_R }
pt_shell> report_case_analysis -sequential_propagation
**************************************
Report : case_analysis
        -sequential_propagation
 ...
**************************************

------------------------------------------------------------
Selective Sequential Propagation Cells : Total = 1
------------------------------------------------------------
Cell Instance : 'FF1'
------------------------------------------------------------
Selective Sequential Propagation Lib Cells : Total = 1
------------------------------------------------------------
Lib Cell : 'snps_test/D_FF'
```

Feedback

See Also

- remove_case_sequential_propagation
- report_case_analysis
- set_case_analysis
- disable_case_analysis
- disable_case_analysis_ti_hi_lo

---

set_cell_mode

Selects the active mode of cell mode groups

Syntax

```
status set_cell_mode
   [mode_list]
   [instance_list]
```

Data Types

```
mode_list          list
instance_list      list
```

Arguments

mode_list

Specifies a list of cell modes, each of which is to be made the active mode for its mode group.

instance_list

Specifies a list of instances for which the specified cell modes are made active. This list must only be included with the *-type cell* value.

Description

This command selects the active mode for a mode group or for several mode groups and disables modes in the same group as the selected active mode. Mode groups must either have all modes enabled (default setting) or have one of their modes enabled and all others disabled. This command sets either cell modes or design modes depending on the value of the type option.

Cell mode groups are defined in the library. Each library cell can have a set of cell mode groups. Each of these cell mode groups can have two or more cell modes. Each of these cell modes can be mapped to a set of timing arcs of the library cell. When a cell mode is made active for a given instance of the library cell, the cell mode is enabled and all of

its timing arcs are enabled for that cell. All other cell modes are automatically disabled. The timing arcs of the disabled cell modes are then automatically disabled. To view what modes have been enabled or disabled use the *report_cell_mode*. To view what arcs have been disabled, use the *report_disable_timing* command.

In the special case of an arc having multiple cell modes mapped to it, the arc is enabled if any of the modes are enabled.

Cell modes can be made active in three different ways, using the *set_cell_mode* command, using the *set_mode -type design* command or through evaluation of mode conditions.

Examples

This command directly selects the READ cell mode as the active mode for the RAM instance Uram1.

```
pt_shell> set_cell_mode READ Uram1
```

See Also

- define_design_mode_group
- map_design_mode
- remove_design_mode
- report_cell_mode
- report_mode
- reset_mode
- set_mode

---

set_check_ctpm_tolerance

Sets the passing threshold used to report passing rate in *gen_ctpm* and *validate_ctpm*.

Syntax

string *set_check_ctpm_tolerance*

```
[-delay float-list]
[-slew float-list]
[-constraint_setup float-list]
[-constraint_hold float-list]
[-constraint_min_pulse_width float-list]
[-delay_sigma float-list]
[-slew_sigma float-list]
[-constraint_setup_sigma float-list]
```

Feedback

```
[-constraint_hold_sigma float-list]
[-pin_cap float-list]
```

Data Types

```
float-list      list of floats
```

Arguments

`-delay float-list`

Specifies the list of delay tolerance values {rel_tol abs_tol}

`-slew float-list`

Specifies the list of slew tolerance values {rel_tol abs_tol}

`-constraint_setup float-list`

Specifies the list of setup constraint tolerance values {rel_tol abs_tol}

`-constraint_hold float-list`

Specifies the list of hold constraint tolerance values {rel_tol abs_tol}

`-constraint_min_pulse_wdith float-list`

Specifies the list of min_pulse_wdith constraint tolerance values {rel_tol abs_tol}

`-delay_sigma float-list`

Specifies the list of delay sigma tolerance values {rel_tol abs_tol}

`-slew_sigma float-list`

Specifies the list of slew sigma tolerance values {rel_tol abs_tol}

`-constraint_setup_sigma float-list`

Specifies the list of setup constraint sigma tolerance values {rel_tol abs_tol}

`-constraint_hold_sigma float-list`

Specifies the list of hold constraint sigma tolerance values {rel_tol abs_tol}

`-pin_cap float-list`

Specifies the list of pin capacitance tolerance values {rel_tol abs_tol}

Description

The *set_check_ctpm_tolerance* command sets the passing threshold used to report passing rate in *gen_ctpm* and *validate_ctpm*. If abs(error percentatage) <= rel_tol or abs(error_in_ns)<abs_tol, the data point is considered as passing. *report_check_ctpm_tolerance* command can be used to report tolerance used.

Feedback

Examples

The following script set delay and slew passing tolerance as 5%/5ps, and report the user-defined tolerance for CTPM checking.

```
set ps_enable_analysis true
set ps_enable_spice2design_analysis true

set_check_ctpm_tolerance -delay {0.05 0.005} -slew {0.05 0.005}
pt_shell> report_check_ctpm_tolerance
  type                          relative_error    absolute_error
  delay                           0.050000            0.005000
  slew                            0.050000            0.005000
  constraint_setup                0.100000            0.010000
  constraint_hold                 0.100000            0.010000
  constraint_min_pulse_width      0.100000            0.010000
  delay_sigma                     0.030000            0.003000
  slew_sigma                      0.030000            0.003000
  constraint_setup_sigma          0.100000            0.010000
  constraint_hold_sigma           0.100000            0.010000
  pin_cap                         0.030000            0.000100
1
```

See Also

- gen_ctpm
- report_check_ctpm_tolerance
- validate_ctpm
- ps_enable_spice2design_analysis

---

set_clock_eco_options

Specifies options for Clock ECO such as *fix_eco_timing* and *fix_eco_drc* command.

Syntax

```
status set_clock_eco_options
   [-from  from_pin]
   [-to to_pin]
   [-levels cell_count]
```

Data Types

```
from_pin          string
to_pin            string
cell_cout         int
```

Feedback

Arguments

`-from` *`from_pin`*

Specifies a clock driver pin or port.

`-to` *`to_pin`*

Specifies a clock load pin or port.

`-levels` *`cell_count`*

Specifies a maximum allowable levels between from_pin and to_pin.

Description

The *set_clock_eco_options* command specifies maximum allowable levels between given from pin and to pin. To run Clock ECO commands such as *fix_eco_timing* and *fix_eco_drc* targeting fixing in clock network. You must use this command before running Clock ECO.

Distributed Multi-Scenario Analysis

To execute the *set_clock_eco_options* command in all scenarios, use the *remote_execute* command. Note that you cannot execute the *set_clock_eco_options* command in the manager process.

Examples

The following example shows setting maximum allowable levels between *U1/Y* and *U2/A* to 3. Suppose there are already 2 cells in between, then *fix_eco_timing\P or fix_eco_drc commands can only insert one buffer between the from_pin and to_pin specified.*

```
pt_shell> set_clock_eco_options -from U1/Y -to U2/A -levels 3
```

The following example shows how to specify the same information in a distributed multi-scenario analysis environment.

```
remote_execute -verbose {
        pt_shell> set_clock_eco_options -from U1/Y -to U2/A -levels 3
}
```

See Also

- fix_eco_drc
- fix_eco_timing

---

set_clock_exclusivity

Specifies a cell for which all the clocks that traverse from the input pins to the output pin will be mutually exclusive.

Feedback

Syntax

```
status set_clock_exclusivity
   -output output_pin
   [-type mux | user_defined]
   [-inputs input_pin_list]
```

Data Types

```
output_pin              list
input_pin_list          list
```

Arguments

`-output output_pin`

This option specifies a single output pin as the exclusivity point. Depending on the *-type* option, all or some of the clocks going out of this pin are considered mutually exclusive.

`-type mux | user_defined`

Specifies the type of clock exclusivity setting, either *mux* or *user_defined*. Use the *mux* setting to set the clock exclusivity for a multiplexer (MUX) cell. Use the *user_defined* setting for a non-MUX cell.

Use the *mux* setting only with the *-output* option, not the *-inputs* option. The specified output pin must belong to a MUX. All clocks at different data signal inputs of the MUX (inputs that are not MUX select pins) are considered mutually exclusive clocks beyond the output. This behavior is similar to setting the *timing_enable_auto_mux_clock_exclusivity* variable to *true*.

You can use the *user_defined* setting for both MUX or non-MUX cells. Use this option with the *-inputs* and *-output* options; the specified input and output pins must belong to the same cell in the design. All clocks at the specified inputs are considered mutually exclusive clocks beyond the output.

`-inputs input_pin_list`

Use this option together with the *-type user_defined* option. The clocks coming through these input pins are exclusive at the output pin specified by the *-output* option.

Description

IC designs often contain MUXes in the clock network to select between multiple exclusive clocks. If the MUX select lines are known not to dynamically change, the clocks going through the input pins of the MUX are in fact exclusive.

In this case, to properly constrain a MUXed clock, we can specify the output pin of cell by using *set_clock_exclusivity* command. The timing paths that are launched from a clock that goes through one input pin and captured by a clock that goes through another input

pin are considered false timing paths. Furthermore, crosstalk interactions between the two clocks coming from different inputs of the exclusive cell are ignored. Please note that if there is a generated clock defined after the exclusivity point, the exclusivity states will be lost. This is similar to the existing behavior of the generated clocks. When a generated clock is created at a pin, all other clocks arriving at that pin are blocked unless they also have generated clock versions created at that pin.

Note that clock exclusivity is not supported by the *extract_model* and *write_eco_design* commands.

HyperScale analysis does consider clock exclusivity. The *characterize_context* command captures clock exclusivity, which is then written out by the *write_context* command. However, note that cross-boundary exclusivity is only written in the binary GBC format; it is not included in the ASCII PTSH constraint format.

To undo the *set_clock_exclusivity* command, use the *remove_clock_exclusivity* command.

To report the exclusivities defined in a design, use the *report_clock* command with the *-exclusivity* option.

Examples

The following example defines a MUX25 cell as a point of exclusivity:

```
pt_shell> set_clock_exclusivity -type mux -output MUX25/Z
```

The following example defines the AND37 gate as a point of exclusivity:

```
pt_shell> set_clock_exclusivity -type user_defined \\
  -output AND37/Z -inputs {AND37/A AND37/B}
```

See Also

- remove_clock_exclusivity
- report_clock
- set_disable_auto_mux_clock_exclusivity
- timing_enable_auto_mux_clock_exclusivity

---

set_clock_gating_check

Specifies the value of setup and hold time for clock gating checks.

Syntax

string *set_clock_gating_check*

```
[-setup setup_value]
[-hold hold_value]
```

```
[-rise | -fall]
[-high | -low]
[-sms_scenarios sms_scenarios_list]
[object_list]
```

Data Types

setup_value	float
hold_value	float
object_list	list
sms_scenarios_list	collection

Arguments

`-setup` *setup_value*

Specifies the clock gating setup time. The default is 0.0.

`-hold` *hold_value*

Specifies the clock gating hold time. The default is 0.0.

`-rise`

Specifies that only rising delays are constrained. By default, if neither the *-rise* or *-fall* options are specified, both rising and falling delays are constrained.

`-fall`

Specifies that only falling delays are constrained. By default, if neither the *-rise* or *-fall* options are specified, both rising and falling delays are constrained.

`-high`

Specifies that the check is performed on the high level of the clock. By default, PrimeTime determines whether to use the high or low level of the clock using information from the cell's logic. That is, for AND and NAND gates PrimeTime performs the check on the high level; for OR and NOR gates, on the low level. For some complex cells (for example, MUX and OR-AND) PrimeTime cannot determine which to use, and does not perform checks unless you specify either the *-high* or *-low* options. If the user-specified value differs from that derived by PrimeTime, the user-specified value takes precedence, and a warning message is issued.

This option sets the attribute only on the specified pin or cell. If you specify the *-high* or *-low* options, you must also specify the *object_list* variable; in that case, the *object_list* variable must not contain a clock.

`-low`

Indicates that the check is performed on the low level of the clock. By default, PrimeTime determines whether to use the high or low clock level using information from the cell's logic. That is, for AND and NAND gates, PrimeTime

performs the check on the high level; for OR and NOR gates, on the low edge. For some complex cells (for example, MUX and OR-AND) PrimeTime cannot determine which to use, and does not perform checks unless you specify either the *-high* or *-low* options. If the user-specified value differs from that derived by PrimeTime, the user-specified value takes precedence, and a warning message is issued.

This option sets the attribute only on the specified pin or cell. If you specify the *-high* or *-low* options, you must also specify the *object_list* variable; in that case, the *object_list* variable must not contain a clock.

object_list

Specifies a list of objects in the current design for which the clock gating check is applied. The objects can be clocks, pins, or cells. If a cell is specified, all input pins of that cell are affected. If a pin or cell is specified, any clock gating checks located at the specified objects are affected. If a clock is specified, the clock gating check is applied to all gating gates driven by that clock. If you specify the *-high* or *-low* options, you must also specify the *object_list* variable; in that case, the *object_list* variable must not contain a clock. By default, if the *object_list* variable is not specified, the clock gating check is applied to the current design.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios for which this clock gating check is applied. When this option is not given the clock gating check applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

The *set_clock_gating_check* command specifies a setup or hold time clock gating check used for clocks, pins, or cells. The gating check is performed on pins that gate a clock signal.

The behavior of the *set_clock_gating_check* command is controlled by the *timing_clock_gating_check_fanout_compatibility* variable.

The clock gating setup check is used to ensure the controlling data signals are stable before the clock is active. This check is performed on combinational gates through which the clock signals are propagated. The arrival time of the leading edge of the clock pin is checked against both levels of any data signals gating the clock. A clock gating setup failure can cause either a glitch at the leading edge of the clock pulse, or a clipped clock pulse.

The clock gating hold check is used to ensure that the controlling data signals are stable while the clock is active. The arrival time of the trailing edge of the clock pin is checked against both levels of any data signal gating the clipped clock pulse.

The *-high* and *-low* options are intended to specify clock gating checks that PrimeTime cannot determine. These options apply only to the pins specified.

To remove information set by *set_clock_gating_check* command, use the *remove_clock_gating_check* command. The *reset_design* command removes all attributes from the design, including those set by the *set_clock_gating_check* command.

To report information for the clock gating check, use the *report_clock_gating_check* command.

Clock gating check constraints are applied in this order:

- Clock gating check set on specific pins or cells
- Clock gating constraint arcs from library or cells on which gating check is set
- Clock gating constraint set on clocks and design

Examples

The following example specifies a setup time of 0.2 and a hold time of 0.4 for all gates in the clock network of clock CK1.

```
pt_shell> set_clock_gating_check -setup 0.2 -hold 0.4 [get_clocks CK1]
```

The following example specifies a setup time of 0.5 on the and1 cell.

```
pt_shell> set_clock_gating_check -setup 0.5 [get_cells and1]
```

See Also

- remove_clock_gating_check
- current_design
- report_constraint
- reset_design
- report_clock_gating_check
- timing_clock_gating_check_fanout_compatibility

---

set_clock_gating_percentage

Sets percentage of clock gating cells to be enabled during vector free vector generation.

Syntax

status *set_clock_gating_percentage*

```
-value percentage_value
[-object_list objecto_list]
```

Data Types

```
percentage_value  float
object_list       list
```

Arguments

`-value percentage_value`

Specifies percentage value for clock gating cells to be enabled. Accepted value is between 0.0 and 1.0, which means 0% to 100% of clock gating cells will be turned on. When this option is specified, it is used to guide vector generation such that the percentage of clock gating cells to be turned on does not exceed this limit. Default is 0.5 which means 50% of clock gating cells will be enabled during vector generation, and the other 50% clock gating cells will be turned off.

`-object_list object_list`

Specifies a list of clock gating cell objects for which the percentage value is applied for. When a clock gating cell is specified with a percentage value, the number of its fanout clock gating cells that can be turned on will be limited by this percentage value. When this option is ommited, the percentage value is applied to the whole design, which means the number of clock gating cells that can be turned on is limited by this percentage value.

Description

This command is used to specify percentage of clock gating cells to be turned on during vector generation in command *write_vectors*.

Examples

The following example sets clock gating cell percentage 20% instead of default 50%, then invokes the command *write_vectors*.

```
pt_shell> set_clock_gating_percentage -value 0.2
pt_shell> write_vectors rtl.vcd
```

Assuming a cascaded clock gating cell structure, clock gating cells icg2, icg3 and icg4 are on the fanout cone of clock gating cell icg1. If icg1 is turned off, none of icg2, icg3 and icg4 can be turned on. If any of icg2, icg3 and icg4 is turned on, that requires icg1 to be turned on first. The following example sets the percentage value 35% on icg1, so at most one of icg2, icg3 and icg4 can be turned on during vector generation.

```
pt_shell> set_clock_gating_percentage -value 0.35 -object_list [get_cell
 icg1]
pt_shell> write_vectors rtl.vcd
```

Feedback

See Also

- write_vectors

---

set_clock_groups

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis.

Syntax

```
status set_clock_groups
   -group clock_list
   [-exclusive]
   [-physically_exclusive]
   [-logically_exclusive]
   [-asynchronous]
   [-allow_paths]
   [-name name]
   [-comment comment_string]
```

Data Types

clock_list	list
comment_string	string
name	string

Arguments

`-group clock_list`

Specifies a list of clocks.

You can use this option multiple times in a single *set_clock_groups* command. Each *-group* option specifies a group of clocks, which are exclusive or asynchronous with the clocks in all other groups. If only one group is specified, it means that the clocks in that group are exclusive or asynchronous with all other clocks in the design. Therefore, a default other group is created for this single group. Whenever a new clock is created, it is automatically included in the default exclusive or asynchronous group.

`-exclusive`

This is an alias for *-logically_exclusive* and is provided for backward compatibility.

`-physically_exclusive`

Specifies that the clock groups are physically exclusive with each other. Physically exclusive clocks cannot coexist in the design physically. An

example of this is multiple clocks that are defined on the same source pin. The *-physically_exclusive*, *-logically_exclusive* and *-asynchronous* options are mutually exclusive; you must choose only one.

`-logically_exclusive`

Logically exclusive clocks do not have any functional paths between them but might have coupling interactions with each other. An example of logically exclusive clocks is multiple clocks, which are selected by a MUX but can still interact through coupling upstream of the MUX cell. The *-physically_exclusive*, *-logically_exclusive* and *-asynchronous* options are mutually exclusive; you must choose only one.

`-asynchronous`

Specifies that the clock groups are asynchronous to each other. Two clocks are asynchronous with respect to each other if they have no phase relationship at all. Signal integrity analysis uses an infinite arrival window on the aggressor unless all the arrival windows on the victim net and the aggressor net are defined by synchronous clocks. The *-physically_exclusive*, *-logically_exclusive* and *-asynchronous* options are mutually exclusive; you must choose only one.

`-allow_paths`

Enable the timing analysis between specified asynchronous clock groups. The default behavior is to suppress timing paths between asynchronous clocks. The *-allow_paths* option must be used with *-asynchronous* option.

`-name` *`name`*

Specifies a name for the clock grouping to be created. Each command should specify a unique name, which identifies the exclusive or asynchronous relationship among specified clock groups, and this name is used later on to easily remove the clock grouping defined here. By default, the command creates a unique name. It takes the first clock name of the first list specified by -group option as the nucleus of the group name. if there is only one group list, then the unique string "_others" is appended. At last, the name is appended with a unique number, starting from 1.

`-comment` *`comment_string`*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

Description

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis. This is similar to using the set_false_path command. In addition, these

relationships also imply the type of crosstalk analysis which should be performed between the clocks. A clock cannot be present in multiple groups during one *set_clock_groups* command execution, but can be included in multiple groups by executing multiple *set_clock_groups* commands. Clock relationships are only implied across the specified clock groups; no relationship is implied across clocks within a group.

For all three relationships, timing paths between the clocks are suppressed. The relationships differ in how crosstalk is handled. If clocks are asynchronous, the clocks are assumed to have an infinite window relationship with each other. If clocks are physically exclusive, only one clock can act as an aggressor or victim during crosstalk analysis; interactions between the clocks are not explored. If clocks are logically exclusive, the crosstalk computation is computed normally (synchronously if the clocks are synchronous) even though the timing paths between the clocks are not reported.

Multiple relationship types can exist between two clocks. When this happens, the relationships obey the following precedence:

- Physically exclusive (highest)
- Asynchronous
- Logically exclusive (lowest)

These precedence rules reflect the real physical behavior of the resulting circuit. For example, if two clocks are asynchronous but they are never simultaneously present, then no crosstalk should be computed.

Note that the traditional *set_false_path* exception between clocks does not result in infinite window crosstalk computation between two clocks. If multiple clocks are asynchronous with each other, the asynchronous relationship should be specified with the *set_clock_groups* command. Failure to specify this relationship for asynchronous clocks could result in an optimistic analysis.

A special *-allow_paths* modifier can be used with the *-asynchronous* option. When used, the clocks are assumed to have an infinite window relationship for crosstalk purposes, but timing paths between the clocks are treated normally. When using this option, paths between the clock domains can be manually disabled using the *set_false_path* command.

When a clock pair is defined such that it has false paths (using either physically or logically exclusive or asynchronous) but subsequently modified to have *-asynchronous -allow_paths*, then this is an error and the first definition holds. In the reverse situation, also the redefinition to false paths from *-allow_paths* between the clock pairs would also be an error. See the UITE-457 and UITE-458 messages.

This command should be used to specify the relationships among clocks before using the *set_active_clocks* command. You should not also set a false path if you have already specified two clocks as exclusive or asynchronous. An error message is issued if you attempt to set a false path between two clocks which are already exclusive or

asynchronous. If a false path was previously set between two clocks when an exclusive or asynchronous relationship is applied, the false path is overwritten by the *set_clock_groups* command. Other exceptions are unaffected.

A clock relationship on a master clock does not automatically propagate to any generated clocks. If the relationship should also apply to generated clocks, they should be explicitly included in the *set_clock_groups* command.

To undo the *set_clock_groups* command, use the *remove_clock_groups* command. To report the clock groups defined in a design, use the *report_clock* command with the *-groups* option.

Examples

The following example defines two asynchronous clock domains:

```
pt_shell> set_clock_groups -asynchronous -name g1 -group CLK33 -group
 CLK50
```

The following example defines a clock named *TESTCLK* to be asynchronous with all other clocks in the design:

```
pt_shell> set_clock_groups -asynchronous -group TESTCLK
```

The following example shows how to simultaneously analyze multiple clocks per register without manually specifying false paths. Assume two pairs of mutually exclusive clocks that are multiplexed:

- CLK1 and CLK2
- CLK3 and CLK4

If each pair of clocks is selected by a different signal, you must execute two *set_clock_groups* commands to specify two independent exclusivity relationships, one for each MUX selection signal:

```
pt_shell> set_clock_groups -physically_exclusive -group CLK1 -group CLK2
pt_shell> set_clock_groups -physically_exclusive -group CLK3 -group CLK4
```

If each pair of clocks is selected by a single MUX selection signal, you would execute only one *set_clock_groups* command to simultaneously analyze all four clocks.

```
pt_shell> set_clock_groups -physically_exclusive -group {CLK1 CLK3}
 -group {CLK2 CLK4}
```

This example does not imply any type of relationship between CLK1-CLK3 or CLK2-CLK4. For example, f CLK1 and CLK3 were also asynchronous with each other, you would need to specify an additional relationship between them.

Feedback

See Also

- create_clock
- create_generated_clock
- get_clock_relationship
- remove_clock_groups
- report_clock
- set_active_clocks
- set_false_path

---

set_clock_jitter

Sets the duty-cycle jitter and/or the cycle clock jitter.

Syntax

```
status set_clock_jitter
   -clock clock_list
   [-cycle cycle_to_cycle_jitter]
   [-duty_cycle duty_cycle_jitter]
```

Data Types

```
clock_list              list
cycle_to_cycle_jitter   float
duty_cycle_jitter       float
```

Arguments

`-clock clock_list`

Specifies a list of clock on which to set the clock jitter.

`-cycle cycle_to_cycle_jitter`

Specifies the cycle-to-cycle jitter on the clock.

`-duty_cycle duty_cycle_jitter`

Specifies the duty-cycle jitter on the clock.

Description

This command specifies the cycle-to-cycle jitter and the duty-cycle jitter on a primary master clock(a clock that is not a generated clock itself), or a generated clock that is defined with the *-multiply_by* option.

Cycle-to-cycle jitter models the variation between the edges that are in-phase. In other words, it models the variation between the edges that are multiple clock cycles apart.

Duty-cycle jitter models the variation between the edges that are out-of-phase. In other words, they are 0.5, 1.5, 2.5, and so on, cycles apart.

Duty-cycle jitter and cycle-to-cycle jitter are realized between the launch clock and the capture clock that are both generated from the same master clock. To figure out the right clock jitter value between the launch clock and capture clock, we trace the launch edge and the capture edge to corresponding edges of the master clock. There are three cases to consider:

1. If the corresponding edges of the master clock are multiple cycles apart. cycle-to-cycle jitter is realized.
2. If they are not and integral number of cycles apart. duty-cycle jitter is realized.
3. If both cycle-to-cycle jitter and duty-cycle jitter are possible. worst jitter among the two is realized.
4. If they are traced back to the same edge of the master clock. no clock jitter is realized.

In crosstalk analysis, clock jitter does not change the arrival windows used in overlap analysis. However, in level-sensitive latch designs, latches can start borrowing in the presence of clock jitter, leading to changes in delta delays for logic nets in the fanout of affected latches.

Please note that the generated clocks defined with the *-multiply_by* option do not inherit the clock jitter from their master clocks.

To remove the clock jitters set by the *set_clock_jitter* command, use the *remove_clock_jitter* command.

To view clock jitter information, use the *report_clock_jitter* command.

Examples

The following example specifies a cycle-to-cycle jitter of 0.5 and duty-cycle jitter of 0.7 on the clock mclk

```
pt_shell> set_clock_jitter -clock [get_clocks mclk] -cycle 0.5
 -duty_cycle 0.7
```

See Also

- remove_clock_jitter
- report_clock_jitter

Feedback

set_clock_latency

Specifies the latency of the clock network.

Syntax

```
status set_clock_latency
[-clock clock_list]
[-rise]
[-fall]
[-min]
[-max]
[-source]
[-dynamic dynamic_component_of_delay]
[-late]
[-early]
[-pll_shift]
[-generated]
[-primary]
[-clock_fall]
[-clock_rise]
[-sms_scenarios sms_scenarios_list]
delay
object_list
```

Data Types

clock_list	list
delay	float
dynamic_component_of_delay	float
object_list	list
sms_scenarios_list	collection

Arguments

`-clock` *clock_list*

Specifies a list of clock objects associated with the network latency placed on all pin/port objects in the *object_list*. Use this option only when the *object_list* contains port/pin objects, not clock objects. To apply a latency value to all objects clocked by a given clock, specify that clock in the *object_list*; do not use the *-clock* option.

`-rise`

Specifies clock rise latency.

`-fall`

Specifies clock fall latency.

`-min`

Specifies clock latency for the minimum operating condition.

`-max`

Specifies clock latency for the maximum operating condition.

`-source`

Specifies clock source latency.

`-dynamic` *`dynamic_component_of_delay`*

Specifies dynamic component of clock latency value.

`-late`

Specifies clock late source latency.

`-early`

Specifies clock early source latency.

`-pll_shift`

Specifies the phase error for output clocks coming out of phase-locked loops (PLLs). This option adds the specified delay to the phase adjustment of the PLL.

`-generated`

Specifies the generated clock signal type. Proper sense should be set on the clock definition point by *set_sense* before applying this option. This option is valid only for clock definition points.

`-primary`

Specifies primary clock signal type (default). Proper sense should be set on the clock definition point by *set_sense* before applying this option. This option is valid only for clock definition points.

`-clock_fall`

Specifies that the signal is coming from the falling edge of the clock. If not specified, it uses the falling edge for *-fall*. Proper sense should be set on the clock definition point by *set_sense* before applying this option. This option is valid only for clock definition points.

`-clock_rise`

Specifies that the signal is coming from the rising edge of the clock. If not specified, it uses the rising edge for *-rise*. Proper sense should be set on the clock definition point by *set_sense* before applying this option. This option is valid only for clock definition points.

delay

Specifies the clock latency value.

object_list

Specifies a list of clocks, pins, or ports for which the latency value applies. If multiple clocks reach these objects, you can restrict the scope of the command to specific clocks by using the *-clock* option.

-sms_scenarios *sms_scenarios_list*

Specifies a collection of SMS scenarios for which this clock latency is applied. When this option is not given the clock latency applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

Two types of clock latency can be specified for a design:

- Clock network latency - This is the time it takes a clock signal to propagate from the clock definition point to a register clock pin. The rise and fall latencies are the latencies for rising and falling transitions at the register clock pin, respectively. Inversion of the clock waveform, if present in the clock network, is not taken into consideration when computing clock network latencies at register clock pins.
- Clock source latency - This is the time it takes for a clock signal to propagate from its actual ideal waveform origin point to the clock definition point in the design. It can be used to model off-chip clock latency when the clock generation circuit is not part of the current design. You can use clock source latency for generated clocks to model the delay from master-clock to generated-clock definition point.

Dynamic clock latency is the component of the total clock latency which is considered dynamic in nature. A dynamic value might differ along successive clock cycles and therefore can only be used to calculate CRP if a zero-cycle path is being considered. A zero-cycle path is one in which the same clock edge both launches and captures.

To specify the dynamic component of any clock latency, use the *-dynamic* option. It can only be specified if the *-source* option and total clock latency is also specified with the command. You can specify dynamic latency on only clock objects, or pin or port objects that are clock sources. When a pin or port object is specified, the clock also has to be specified by *-clock* option.

By default, the tool assumes ideal clocking, which means clocks have a specified network latency, designated by the *set_clock_latency* command, or zero network latency if no latency value is specified. Propagated clock network latency, designated by the *set_propagated_clock* command, is normally used after layout after final clock tree generation. Ideal clock network latency provides an estimate of the clock tree for pre-layout.

You can specify clock source latency for ideal or propagated clocks. The total clock latency at a register clock pin is the sum of clock source latency and clock network latency. If the rise and fall latencies are different, the source latencies for a latch are chosen based on the sense at the clock port and network latencies are chosen based on the sense at the latch clock pin. Thus, source latency takes into account the clock inversions on the clock network, but network latency does not.

In the on_chip_variation_analysis mode, the min/max operating conditions are the variation bounds within the single corner analysis. Therefore, only the min-early and max-late latencies are used for the analysis; it is sufficient to specify either the *-early* or *-late* options or the *-min* or *-max* options for the *set_clock_latency* command.

For internally generated clocks, the tool automatically computes the clock source latency if the master clock of the generated clock has propagated latency and there is no user-specified value for the generated clock source. Zero latency is assumed if the master clock is ideal, the master clock has source latency, and there is no user-specified value for the generated clock's source latency.

If the *set_clock_latency* command is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports. Directly setting a network latency makes the affected registers ideal. A warning is issued if the *set_propagated_clock* command has previously set a propagated attribute on the same object (clock, pin, or port). Clock source latencies are allowed only on clocks and clock source pins.

The command does not check to see whether a clock specified with the *-clock* option passes through the pin or pins referenced in the object list. If the clock specified does not pass through the pin, the command has no effect and there is no warning about this condition.

To undo the *set_clock_latency* command, use the *remove_clock_latency* command.

To see clock network and source latency information (including dynamic component), use the *report_clock* command with the *-skew* option.

Examples

The following example specifies a rise latency of 1.2 and a fall latency of 0.9 for the CLK1 clock.

```
pt_shell> set_clock_latency 1.2 \\
          -rise [get_clocks CLK1]
pt_shell> set_clock_latency 0.9 \\
          -fall [get_clocks CLK1]
```

The following example specifies an early rise and a fall source latency of 0.8 and a late rise and fall source latency of 0.9 for the CLK1 clock.

```
pt_shell> set_clock_latency 0.8 -source \\
          -early [get_clocks CLK1]
```

```
pt_shell> set_clock_latency 0.9 -source \\
          -late  [get_clocks CLK1]
```

The following example specifies an early and late source latency of 3 and 5, respectively, with dynamic components of 0.5.

```
pt_shell> set_clock_latency 3 -source \\
          -early -dynamic 0.5 [get_clocks CLK1]
pt_shell> set_clock_latency 5 -source \\
          -late  -dynamic 0.5 [get_clocks CLK1]
```

The following example specifies the source latency as well as the dynamic portion of source latency on a clock source pin.

```
pt_shell> create_clock -name clk -period 10 [get_pins clk_source_pin]
pt_shell> set_clock_latency 100 -source [get_pins clk_source_pin] \\
          -dynamic 20 -clock [get_clocks CLK1]
```

The following example specifies generated clock source latencies of 0.5 to the rise-to-rise clock edge and 0.6 to the fall-to-fall clock edge on pin and2/Y.

```
pt_shell> set_clock_latency 0.5 -source \\
          -generated -rise -clock_rise and2/Y
pt_shell> set_clock_latency 0.6 -source \\
          -generated -fall -clock_fall and2/Y
```

The following example specifies primary clock source latencies of 0.1 to the rise-to-rise clock edge and 0.2 to the fall-to-fall clock edge on pin and2/Y.

```
pt_shell> set_clock_latency 0.1 -source \\
          -primary -rise -clock_rise and2/Y
pt_shell> set_clock_latency 0.2 -source \\
          -primary -fall -clock_fall and2/Y
```

The following example shows a timing report where dynamic clock latency has been specified with the *set_clock_latency* command:

```
pt_shell> report_timing -path_type full_clock_expanded
 ...

  Startpoint: ff1 (rising edge-triggered flip-flop clocked by clk)
  Endpoint: ff2 (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock source latency                     3.00       3.00
  clock source latency (dynamic)           0.60       3.60
  clk (in)                                 0.00       3.60 r
  u1/Z (IBUF1)                             0.85       4.45 r
```

```
ff1/CP (FD1)                                0.00       4.45 r
ff1/Q (FD1)                                 1.44       5.89 f
u3/Z (IV)                                   0.58       6.47 r
ff2/D (FD1)                                 0.00       6.47 r
data arrival time                                      6.47

clock clk (rise edge)                      10.00      10.00
clock source latency                        1.00      11.00
clock source latency (dynamic)             -0.50      10.50
clk (in)                                    0.00      10.50 r
u1/Z (IBUF1)                                0.85      11.35 r
u2/Z (IBUF1)                                0.68      12.03 r
ff2/CP (FD1)                                0.00      12.03 r
clock reconvergence pessimism               2.00      14.03
library setup time                         -0.80      13.23
data required time                                    13.23
-------------------------------------------------------------
data required time                                    13.23
data arrival time                                     -6.47
-------------------------------------------------------------
slack (MET)                                            6.76
```

See Also

- remove_clock_latency
- report_clock
- set_clock_transition
- set_clock_uncertainty
- set_propagated_clock

---

set_clock_map

Defines the mapping of clocks across hierarchical design levels for HyperScale analysis, overriding the default inferred clock mapping.

Syntax

status *set_clock_map*

```
[-parent_clocks parent_clock_names]
[-block_clocks block_clock_names]
[-instances inst_cells]
[-add]
clock_list
```

Feedback

Data Types

parent_clock_names	list
block_clock_names	list
inst_cells	list
clock_list	list

Arguments

`-parent_clocks` *parent_clock_names*

Specifies the names of the clocks used at the parent hierarchy level that are the same as the clocks listed in the *clock_list*. This option is valid only for block-level analysis and cannot be used with the *-block_clocks* option.

`-block_clocks` *block_clock_names*

Specifies the names of the clocks used at the subblock level that are the same as the clocks identified in the *clock_list*. This option is valid only for analysis at a higher level of hierarchy and cannot be used with the *-parent_clocks* option.

`-instances` *inst_cells*

Specifies a list of hierarchical subblock instances in the current design for which the clock mapping applies. Clock mapping can be instance-specific even when different instances refer to same physical block. This option can be used only with the *-block_clocks* option.

`-add`

Adds this clock mapping to the default inferred clock associations. Without this option, you take full control of the clock mapping to the parent level or block level; the new clock mapping overwrites any existing clock mappings inferred by the tool. This option does not affect clock mappings defined by previous *set_clock_map* commands; such mappings remain in effect until you use the *remove_clock_map* command.

clock_list

Specifies a list of collections containing clocks or patterns matching the clock names in the current design. The command maps these clocks to the parent clock named by the *-parent_clocks* option or to the lower-level clocks named by the *-block_clocks* option.

Description

In a hierarchical design flow, top-level and block-level designs often need to be implemented and analyzed separately, with separate but highly associated block-level and top-level timing constraints. The same clocks might be assigned different names at different hierarchical levels.

The PrimeTime tool automatically detects and associates clocks with mismatching names across hierarchical levels by considering the clock characteristics. In the HyperScale flow, the tool usually establishes the clock mapping correctly, especially when you use HyperScale constraint extraction.

However, in complex or ambiguous clocking situations with multiple clocks, using the available design information, the tool cannot always associate these clocks as intended. To explicitly specify the mapping and override the default inferred mapping, use the *set_clock_map* command with the appropriate options:

- To define the mapping of clocks at the current level with clocks at the parent level, use the *-parent_clocks* option.
- To define the mapping of clocks at the current level with clocks at the subblock level, use the *-block_clocks* option, and optionally use the *-instances* option to apply the mapping to specific instances.

You cannot use the *-parent_clocks* and *-block_clocks* options at the same time. To define both parent and subblock mappings for an intermediate-level design, use two separate *set_clock_map* commands.

To remove clock mappings set with the *set_clock_map* command, use the *remove_clock_map* command.

Reporting the Clock Mapping

To show the clock mappings used by the tool for timing analysis, use the *report_clock -map* command. The report shows an asterisk next to each clock mapped explicitly by the *report_clock -map* command:

```
pt_shell> set_clock_map MCLK -block_clocks SCLK
Information: Abandoning fast timing updates. (PTE-018)
1
pt_shell> update_timing -full
Information: Loading block timing data for block instance low_1. (HS-001)
Information: applying user mapping between block clock 'SCLK' and clock
 'MCLK'.
Information: Loading block timing data for block instance low_1. (HS-001)
1
pt_shell> report_clock -map
 ...
Instance          Top level clock     Block level clock
-------------------------------------------------------------
low_1             (N/A)               MCLK
                  MCLK                SCLK*
```

Situations That Require Explicit Clock Mapping

The most common type of ambiguity is caused by clocks with same period and waveform propagating though overlapping networks across a block boundary. For example, multiple

top-level clocks with same basic period and waveform characteristics can all reach a single block port and enter into a block. In this case, block-level constraints have multiple clocks defined on the port to represent all the top clocks. Explicitly defining the mapping with the *set_clock_map* command is the most reliable way to ensure correct mapping.

Another common ambiguity is caused by virtual clocks, which are used to model a clock signal without a physical source (and network) present in the current level of design. For example, even when a top-level clock does not physically propagate into a specific hierarchical block itself, it can still launch data signals that enter the block or capture signals that leave the block. To model such a clock in block-level constraints, you can define a virtual clock with a period and waveform matching the top-level physical clock, but without attaching it to any source or source network objects in the block design. Ambiguity arises when multiple top-level clocks have the same characteristics. Use the *set_clock_map* command to resolve such ambiguities.

You can choose to overdesign at the block level by defining clocks with tighter margins or higher frequencies than the same clocks at the top level. When the clock characteristics do not match exactly between the block-level and top-level clocks, the tool does not establish mapping between the clocks by default because of the clock differences. To override this default behavior, use the *set_clock_map* command to explicitly map the slightly mismatching clocks. In this situation, the *report_constraint* command still reports the mismatching clocks as a boundary constraint violations.

You can modify the behavior of automatic clock mapping by setting the variables *hier_disable_auto_clock_mapping* and *hier_clock_mapping_period_percent_tolerance*.

Clock Mapping and HyperScale Technology

Use the *set_clock_map* command in a manner consistent with the HyperScale configuration command *set_hier_config*. For example, for block-level analysis, a clock is defined with the name BCLK, which represents a top-level counterpart with the name TCLK. (Note that TCLK is defined in a separate set of top-level constraints and is not included in the constraints for block-level analysis.) Use the following commands at the block level:

```
set_hier_config -parent -path $top_dir ...
 ...
create_clock -name BCLK ...
 ...
set_clock_map BCLK -parent_clocks TCLK
 ...
update_timing
 ...
```

For top-level analysis, the clock is defined with name TCLK:

```
set_hier_config -block A_BLK -path $blk_dir ...
 ...
create_clock -name TCLK ...
```

```
 ...
set_clock_map TCLK -block_clocks BCLK
 ...
update_timing
 ...
```

Mapping a Top-Level Clock to Multiple Block-Level Clocks

HyperScale technology supports one-to-one, one-to-N, and N-to-one clock mappings between the top-level clocks and block-level clocks.

Most often, with a top-level clock propagating into a block through a single port, a single block-level clock is defined on the port and uniquely maps to the top-level clock. However, there are cases a single block-level clock definition cannot fully represent the top-level clock intent. For example:

- Multiple waveforms from the same clock -- The top-level clock enters into a block through multiple ports with a positive waveform on some ports but inverted through others. This requires block-level constraints to define two clocks with opposite waveforms on the associated ports.
- Re-entrant generated clocks -- This happens when a generated clock defined inside a block propagates outside of the block and re-enters the same block through some port. A generated clock requires the source network path back to its master clock to have a valid source latency computation. The portion of the source network traveling outside of the block does not exist for block-level analysis, so block-level constraints need to define two separate clocks to fully represent the single generated clock in top-level constraints.

By default, the tool automatically establishes 1-to-N clock mapping for most of the these configurations. However, if the design is too complex or ambiguous for reliable automatic mapping, you should use the *set_clock_map* command to explicitly specify the mapping.

Examples

The following example defines the mapping of top-level clock SYSCLK to block-level clock BCLK in HyperScale top-level analysis.

```
# Map SYSCLK at current level to BCLK in HyperScale block
set_clock_map SYSCLK -block_clocks BCLK
```

The following example defines the mapping of top-level clock SYSCLK to block-level clock BCLK_1 for one instance and BCLK_2 for another instance.

```
set_clock_map SYSCLK -block_clocks BCLK_1 -instance blk_inst1
set_clock_map SYSCLK -block_clocks BCLK_2 -instance blk_inst2
```

In the following block-level analysis example, two block-level clocks map to the same top-level clock.

```
set_clock_map sysCLK_pos -parent_clocks SYSCLK
set_clock_map sysCLK_neg -parent_clocks SYSCLK
```

The following example shows the clock mapping reported with both automatically resolved and user-specified explicit clock mappings. User-specified mappings are annotated with an asterisk (*) in the report.

```
pt_shell> report_clock -map
 ...
Instance        Top level clock     Block level clock
-----------------------------------------------------
core3           CLK1                CLK1* CLK1_v*
                CLK2                CLK2  CLK2_v
                CLK3                CLK3  CLK3_v
                CLK4                CLK4* CLK4_v*
core4           CLK1                CLK1* CLK1_v*
                CLK2                CLK2  CLK2_v
                CLK3                CLK3  CLK3_v
                CLK4                CLK4  CLK4_v
core1           CLK1                CLK1* CLK1_v*
                CLK2                CLK2* CLK2_v*
                CLK3                CLK3  CLK3_v
                CLK4                CLK4  CLK4_v
core2           CLK1                CLK1* CLK1_v*
                CLK2                CLK2* CLK2_v*
                CLK3                CLK3  CLK3_v
                CLK4                CLK4  CLK4_v
```

See Also

- create_clock
- create_generated_clock
- remove_clock_map
- report_clock
- report_constraint
- set_hier_config
- clock_mapping_violations
- hier_clock_mapping_period_percent_tolerance
- hier_disable_auto_clock_mapping

set_clock_sense

The *set_clock_sense* command is deprecated. Use the *set_sense* command instead.

Feedback

See Also

- remove_sense
- set_sense

---

set_clock_transition

Specifies transition time of register clock pins.

Syntax

string *set_clock_transition*

```
[-rise]
[-fall]
[-min]
[-max]
transition
clock_list
```

Data Types

transition	float
clock_list	list

Arguments

`-rise`

Specifies clock transition time for rising clock edge.

`-fall`

Specifies clock transition time for falling clock edge.

`-min`

Specifies clock transition time for minimum conditions.

`-max`

Specifies clock transition time for maximum conditions.

transition

Specifies transition time of clock pins.

clock_list

Provides a list of clocks in the design. The transition times on all register clock pins and clock gating constraint arcs in the transitive fanout of specified clock are affected. You only can specify clocks with ideal latency in the list. When

affected clock pins in the fanout of a clock are marked with propagated latency, the values are ignored in the *set_clock_transition* command.

Description

This command provides the ability to override the calculated transition times on clock pins of registers and clock gating constraint arcs.

This command is useful for pre-layout when clock trees are incomplete and using calculated transition times at register clock pins and for clock gating constraint arcs can be highly pessimistic. The transition value specified with this command overrides the transition times of all nets directly feeding a sequential element clocked by the specified clock.

Use the *set_clock_transition* command only with ideal clocks. For propagated clocks the calculated transition times are used. Propagated clocks are only set after the final clock tree is constructed.

If a clock transition is not specified for an ideal clock, and if the timing_ideal_clock_zero_default_transition variable is TRUE (by default it is TRUE), then zero transition is used. Otherwise, the transition time is calculated as it is for other pins in the design.

To undo the *set_clock_transition* command, use the *remove_clock_transition* command.

To list all clock transition values which have been set, use *report_clock -skew*.

Examples

This example specifies a rise transition time of 0.38 and fall transition time of 0.25 for register clock pins clocked by "CLK1".

```
pt_shell> set_clock_transition 0.38 -rise [get_clocks CLK1]
pt_shell> set_clock_transition 0.25 -fall [get_clocks CLK1]
```

See Also

- remove_clock_transition
- report_clock
- set_clock_latency
- set_clock_uncertainty
- set_propagated_clock
- timing_ideal_clock_zero_default_transition

Feedback

set_clock_uncertainty

Specifies the uncertainty (skew) of specified clock networks.

Syntax

string *set_clock_uncertainty*

```
[-from from_clock]
[-rise_from rise_from_clock]
[-fall_from fall_from_clock]
[-to to_clock]
[-rise_to rise_to_clock]
[-fall_to fall_to_clock]
[object_list]
[-rise]
[-fall]
[-setup]
[-hold]
[-sms_scenarios sms_scenarios_list]
uncertainty
```

Data Types

```
fall_from_clock      list
fall_to_clock        list
from_clock           list
object_list          list
rise_from_clock      list
rise_to_clock        list
sms_scenarios_list   collection
to_clock             list
uncertainty          float
```

Arguments

`-from from_clock`

Specifies the source clock for interclock uncertainty.

You must specify either simple uncertainty using an object list or interclock uncertainty using a combination of *-from*, *-rise_from*, or *-fall_from* and *-to*, *-rise_to*, or *-fall_to*. You cannot specify both simple and interclock uncertainty in the same command.

`-rise_from rise_from_clock`

Specifies the source clock for interclock uncertainty for only rising edges of the clock.

`-fall_from` *`fall_from_clock`*

Specifies the source clock for interclock uncertainty for only falling edges of the clock.

`-to` *`to_clock`*

Specifies the destination clock for interclock uncertainty.

`-rise_to` *`rise_to_clock`*

Specifies the destination clock for interclock uncertainty for only rising edges of the clock.

`-fall_to` *`fall_to_clock`*

Specifies the destination clock for interclock uncertainty for only rising edges of the clock.

`object_list`

Specifies a list of clocks, ports, or pins for simple uncertainty. For a clock object, the uncertainty applies to capturing latches clocked by that clock. For a port or pin, the uncertainty applies to capturing latches whose clock pins are in the fanout of that port or pin, including transitive fanout through combinational logic. This option cannot be used with the "from" and "to" type options.

`-rise`

Applies interclock uncertainty only to the rising edge of the destination clock. This option is supported only for backward compatibility; use the *-rise_to* option instead.

`-fall`

Applies interclock uncertainty only to the falling edge of the destination clock. This option is supported only for backward compatibility; use the *-fall_to* option instead.

`-setup`

Applies the uncertainty only to setup checks. By default, the uncertainty applies to both setup and hold checks.

`-hold`

Applies the uncertainty only to hold checks. By default, the uncertainty applies to both setup and hold checks.

`uncertainty`

The clock uncertainty, a positive number representing the amount of variation in arrival times of successive clock edges, in library time units.

Negative uncertainty values are supported for constraining designs with complex clock relationships. However, setting a negative value could lead to optimistic analysis and should be done with extreme care.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this specification is applied. When this option is not given, the specification applies to all SMS scenarios. This collection is created by the *get_sms_scenarios* command.

Description

The *set_clock_uncertainty* command specifies the clock uncertainty (skew) of specified clock networks. This is the amount of variation in clock edge arrivals away from the exact times expected for a perfect clock.

Set the uncertainty to the worst skew expected to the endpoint or between the clock domains. You can increase this value get additional margin for setup and hold checking.

When calculating slack, the tool adjusts the required data arrival time to make the check more restrictive, thus considering the worst possible variation in arrival times for successive clock edges.

You can specify either interclock or simple uncertainty:

- For interclock uncertainty (between two different clocks), use a combination of the *-from*, *-rise_from*, or *-fall_to* option to specify the source (launch) clock and the *-to*, *-rise_to*, or *-fall_to* option to specify the destination (capture) clock, as well as the applicable edge types (rising, falling, or both). The interclock uncertainty value applies to all paths launched and captured by the respective specified clocks.
- For simple uncertainty (between successive edges of the same clock), specify an object list containing one or more clocks, ports, or pins to which the uncertainty applies. For a clock, the uncertainty applies to all capturing devices clocked by that clock. For a port or pin, the uncertainty applies to capturing devices whose clock pins are in the fanout of that port or pin.

Be sure to specify interclock uncertainty for all possible interactions of clock domains. For example, if you have paths from CLKA to CLKB and from CLKB to CLKA, you need to specify the uncertainty for both combinations, even if the values are the same.

In crosstalk analysis, clock uncertainty does not change the arrival windows used in overlap analysis. However, in level-sensitive latch designs, latches can start borrowing in the presence of clock uncertainty, leading to changes in delta delays for logic nets in the fanout of affected latches.

Feedback

Conflicting uncertainty settings have the following order of priority, from highest to lowest:

- Interclock uncertainty set with *-from* and *-to* type options
- Simple uncertainty set on ports or pins
- Simple uncertainty set on clocks

If you set simple uncertainty on a clock port or pin, the uncertainty applies to capturing latches whose clock pins are in the transitive fanout of the specified object. If different uncertainty values are propagated from multiple ports or pins to inputs of a clock MUX, the worst uncertainty setting applies to the entire fanout of the MUX, even when the associated clock is disabled at the MUX. To ensure usage of the proper uncertainty value in this situation, set the uncertainty on the clock object rather than on a clock port or clock pin.

To view clock uncertainty settings, use the *report_clock -skew* command.

To remove uncertainty settings, use the *remove_clock_uncertainty* command.

Examples

The following commands set the simple uncertainty to 0.65 for setup checking and 0.45 for hold checking for all paths leading to registers or ports clocked by clock CLK.

```
pt_shell> set_clock_uncertainty -setup 0.65 [get_clocks CLK]
pt_shell> set_clock_uncertainty -hold 0.45 [get_clocks CLK]
```

The following commands specify the interclock uncertainty values between the PHI1 and PHI2 clock domains.

```
pt_shell> set_clock_uncertainty 0.4 -from PHI1 -to PHI1
pt_shell> set_clock_uncertainty 0.4 -from PHI2 -to PHI2
pt_shell> set_clock_uncertainty 1.1 -from PHI1 -to PHI2
pt_shell> set_clock_uncertainty 1.1 -from PHI2 -to PHI1
```

Note that both PHI1 to PHI2 and PHI2 to PHI1 must be specified, even though it is the same uncertainty value. Each uncertainty setting applies only in the specified direction, from source to destination.

The following commands specify interclock uncertainty values between PHI1 and PHI2 clock domains for specific rise/fall edge types.

```
pt_shell> set_clock_uncertainty 0.4 -rise_from PHI1 -to PHI2
pt_shell> set_clock_uncertainty 0.4 -fall_from PHI2 -rise_to PHI2
pt_shell> set_clock_uncertainty 1.0 -from PHI1 -fall_to PHI2
```

The following commands set both simple and interclock uncertainty. Where the settings are in conflict, the interclock uncertainty setting (2.0) has higher priority.

```
pt_shell> set_clock_uncertainty 4.0 [get_clocks CLKA]
pt_shell> set_clock_uncertainty 2.0 -from CLKB -to CLKA
```

See Also

- remove_clock_uncertainty
- report_clock
- set_clock_latency
- set_clock_transition
- timing_include_uncertainty_for_pulse_checks
- timing_propagate_interclock_uncertainty

---

set_command_option_value

Set a command option default or current value.

Syntax

string *set_command_option_value*

-default | -current

-command *command_name*

-option *option_name* | -position *positional_option_position*

-value *value* | -undefined

Arguments

`-default`

Set the default option value.

`-current`

Set the current option value.

`-command` *`command_name`*

Set the option value for an option of this command.

`-option` *`option_name`*

Set the option value for this option of the command.

`-position` *`positional_option_position`*

Set the option value for this positional option of the command.

`-value value`

Set the option value to this value.

`-undefined`

Set the option value to the "undefined value".

Description

Sets the current or default value of a command option.

Either -default must be specified (to specify setting the default value of the option), or -current must be specified (to specify setting the current value of the option). Unlike for get_command_option_values, one of -current or -default must be specified for set_command_option_value.

An option of a command must be specified (for value setting) by passing a command name via -command and either an option name (for an option of the command) via -option, or the position of a positional option via -position. The *option_name* passed via -option (for a dash option) must not have its leading dash character omitted. An option name passed via -option may be either the name of a dash option or the name of a positional option. A positional option may be specified either via -option or via -position. Positional option positions of a command are numbered 0, 1, 2, ... (N-1), where N is the number of positional options of the command.

To set the (current or default) value of the option, a string value should be passed via -value. Alternatively, you can pass -undefined to reset the option to have the "undefined value". (CAVEAT: For some "set value implementations" for numeric options, -undefined does not really "undefine" the value - instead the value is set to 0.)

Processing of the -value value does not include any CCI option checking of the value (such as checking whether the option value has correct syntax for its type).

The command "throws" a Tcl error for various conditions:

* the command does not exist

* the command exists but no such option of the command exists

* the set operation failed (could be due to a failed conversion for one of the "set value" implementations which does a conversion from string to one of integer, double, etc.)

The -undefined option is a mutually exclusive alternative to the -value option. -undefined has the effect of resetting the option value to the "undefined value".

A possible power user application: power users could put set_command_option_value commands in their home directory Tcl startup files for their favorite dialogs/commands to "seed" initial current values for these dialogs. This may lower the need for automatically saving replay scripts (of set_command_option_value commands for each command/dialog option/field) on exit from the application.

Feedback

Examples

The following example sets the current value for the -bar option of the foo command to a value of abc.

```
prompt> set_command_option_value -current -command foo \\
   -option "-bar" -value abc
```

The following example sets the current value for the first positional argument of the foo command to a value of xyz.

```
prompt> set_command_option_value -current -command foo \\
   -position 1 -value xyz
```

See Also

- get_command_option_values
- preview

---

set_concurrent_event_analysis_options

Sets the options for concurrent event analysis in PrimePower.

Syntax

integer *set_concurrent_event_analysis_options*

```
[-max_process number]
[-partition_by_design ]
[-merge_output_waveform]
[-merge_activity_waveform]
[-keep_overlapped]
```

Data Types

number	integer

Arguments

`-max_process`

Specifies the maximum number of processes to be used for concurrent event analysis.

`-partition_by_design`

When this option is specified , design partitioning is used instead of FSDB time partitions. The design is partitioned into multiple sub designs and each process runs a sub design. This method is suitable when the design size is huge and processing the entire design in a given machine may not be feasible due to memory constraints. This option is supported only for time-based mode.

`-merge_output_waveform`

Merge output fsdb waveform generated inside partitions into single fsdb waveform.

IP -merge_activity_waveform Merges the output activity waveforms generated inside partitions into single activity waveform.

`-keep_overlapped`

This can be used to smoothen the merged power waveform if it has holes at the borders of partition windows.

Description

This command specifies the options for concurrent event analysis. A maximum of 16 processes are allowed for concurrent event analysis . Depending on the FSDB data on the given time window, PrimePower may be choose less number of processes. If this command is not specified , PrimePower uses 4 processes for concurrent event analysis. Increasing the number of processes can speedup the runtime at the cost of more memory .

If you are setting the max_process value above 4 , it is recommended to increase the multi-core limit of the tool by using *'set_host_options -max_cores'* at the beginning of the script . This will ensure maximum runtime benefit for concurrent event analysis .

Examples

The following example will cause PrimePower to do concurrent event analysis with upto 16 processes.

```
pt_shell> set power_enable_concurrent_event_analysis true
pt_shell> set_concurrent_event_analysis_options -max_process 16
pt_shell> update_power
```

See Also

- power_enable_concurrent_event_analysis
- set_host_options

---

set_connection_class

Sets the connection class value on ports.

Syntax

int *set_connection_class*

```
connection_class_value
object_list
```

Feedback

Data Types

connection_class_value	string
object_list	list

Arguments

connection_class_value

Specifies the desired *connection_class* value of ports contained in the *object_list* option. The *connection_class_value* option is a single string value. This string can contain any number of space separated connection classes (see the example below).

object_list

Specifies ports whose connection classes are set.

Description

The *connection class* label is an attribute that is used to describe connection requirements for a given technology. Only those loads and drivers with the same connection class label can be legally connected. The labels "universal" and "default" are reserved labels. The "universal" label indicates that a pin or port can legally connect with any other load or driver. The "default" label is used for those library pins that do not otherwise have a connection class assigned to them through any library attributes. For designs being optimized analyzed, the "universal" label is assigned to those ports that do not otherwise have a connection class assigned to them. To change the default connection class label for those ports, use the *default_port_connection_class* variable.

Connection classes are placed on ports of designs in a technology library by using Library Compiler. The *set_connection_class* command places connection classes on ports of designs being analyzed (such as the current design). This is used to indicate a connection class requirement for the network that is connected to the specified port.

To view connection class values on ports, use the *get_attribute* command. To check your design for connection class violations, use the *report_constraint* command.

Examples

The following example sets a connection class "internal" to the port named "xyz" in the current design.

```
pt_shell> set_connection_class  internal xyz
```

The following example sets the connection classes "internal" and "external" on the port "out" in the design "test".

```
dc_shell> set_connection_class "internal
                          external" test/out
```

In the following example, the connection class on a port is removed.

```
pt_shell> remove_connection_class  test/out
```

See Also

- remove_connection_class
- report_constraint

---

set_context_margin

Adds delay margin to the block context data written by the *write_hier_data* and *write_context* commands, resulting in a more conservative hierarchical analysis.

Syntax

status *set_context_margin*

```
[-max]
[-min]
[-rise]
[-fall]
[-type clock | data]
[-allocation pin_slack | clock_period | user_override]
[-percent]
value
[objects]
```

Data Types

value	double
objects	list

Arguments

`-max`

Indicates that the specification applies only to max-delay delays at block boundaries.

`-min`

Indicates that the specification applies only to min-delay delays at block boundaries.

`-rise`

Indicates that the specification applies only to rising-edge delays at block boundaries.

`-fall`

Indicates that the specification applies only to falling-edge delays at block boundaries.

`-type clock | data`

Modifies the delay constraints only for the specified type of paths cross the block boundary, either *clock* (*set_clock_latency* commands) or *data* (*set_input_delay* and *set_output_delay* commands). By default, both types of delay constraints are affected.

`-allocation pin_slack | clock_period | user_override`

Modifies delay constraints using the specified method:

- *pin_slack* - Allocates a percentage (specified by the *-percent* option) of the delay slack to the block level and the remaining portion to the top level. The top-level portion of the slack is added to or subtracted from the actual delay at the top level to or from the pin, thereby budgeting that portion of the slack to the context surrounding the block. You can use this option only with the *-type data* option.
- *clock_period* - Sets the delay value to the percentage of the clock period specified by the *-percent* option. The actual delay at the top level is ignored.
- *user_override* - Sets the delay to exactly the value specified by the *value* argument of the command. The actual delay at the top level is ignored. Do this with caution, as relaxed delay values cause optimism.

If you do not use the *-allocation* option, the command adds the *value* argument to (or subtracts the *value* argument from) the delay values of the *set_input_delay*, *set_output_delay*, or *set_clock_latency* commands written out for the block context.

`-percent`

Specifies percentage by which the context delay value is modified, or specifies the percentage value for the *-allocation clock_period* or *-allocation user_override* option. Using the *-percent* option is mandatory for the *-allocation clock_period* or *-allocation pin_slack* option.

When using the *-max* and *-percent* options (without the *-allocation* option), specify a *value* of 10.0 for a 10 percent increase in the delays specified by the *set_input_delay* and *set_output_delay* commands written for the block context.

Without the *-percent* option, by default, the *value* argument is an absolute amount added to, or subtracted from, the actual delay value for the context.

objects

Specifies where to add the block context margin.

In a top-level analysis, you can specify a list of block instance cells or instance pins. If you omit the *objects* argument, the margin applies to all block instances. The margin is included in the block context data written out by the *write_hier_data* command.

In a block-level analysis, you can specify a list of ports. The margin is applied to the block context data read in by the *read_context_data* command. If the context data already contains margins from the top level, they are combined additively.

Port objects cannot be specified with the following options: *-allocation pin_slack*, *-allocation clock_period*, *-percent*.

value

Specifies the margin value to be added to (or subtracted from) delay values used in block context data.

If you specify a positive value, both min-delay and max-delay values are modified to tighten the constraint and increase conservatism. To relax rather than tighten the constraint check, specify a negative value. The units are the applicable time units of the logic library.

If the *-percent* option is used, the *value* specifies the applicable percentage, from 0.0 to 100.0 percent.

If the *-allocation user_override* option is used, the *value* specifies the time value of the delay constraint.

Description

The *set_context_margin* command tightens or relaxes clock/data delay values by a specified amount in the block context data written or read by the following commands:

- *write_context*
- *write_hier_data* (in a HyperScale top-level run)
- *read_context_data* (in a HyperScale block-level run)

If the top-level logic is still uncertain, this margin can help improve future convergence between block-level and top-level timing analysis runs.

The specified margin modifies the delay values of the following commands in the block context data:

```
set_input_delay
set_output_delay
set_clock_latency -source
```

A positive margin *value* tightens both min-delay and max-delay analysis. (In the *set_input_delay* and *set_output_delay* commands in the block context, the margin is added to *-max* delay values and subtracted from *-min* delay values.)

A negative *value* relaxes the constraints instead of tightening them. Do this with caution because it introduces optimism.

Using the *set_context_margin* command triggers a full timing update. To reduce runtime, use the command early in the analysis flow, before the *update_timing* or *report_timing* command.

The *set_context_margin* command offers the following options:

- The amount and direction of the delay constraint adjustment
- The adjustment method: absolute adjustment, delay percentage, slack percentage, total delay as a percentage of the clock period, or total delay in time units
- Whether to apply the delay change to maximum or minimum constraints only, to rise or fall transitions only, or to data or clock paths only (default is all)
- In a HyperScale top-level run, whether to apply the delay change only to specific blocks or block pins, or to all blocks for which the context is being generated (the default)
- In a HyperScale block-level run, which ports the delay change should apply to

To cancel a margin setting made previously by the *set_context_margin* command, use the *remove_context_margin* command.

These are the available adjustment methods:

- Absolute adjustment (added to or subtracted from the actual delay)
- Adjustment as a percentage of the actual delay
- Adjustment as a percentage of the pin slack
- Total delay as a percentage of the clock period
- Total delay in time units

Relative Adjustment in Time Units

The *write_context* command writes out the block context as *set_input_delay* commands that specify the delays from data launch to arrival at the block inputs, and *set_output_delay* commands that specify the delays from block outputs to data capture. These commands provide the constraints for block timing analysis in isolation from the top level.

To adjust these delays by a specified amount of time, use the *set_context_margin* command with the *value* option. Specify a positive value for a more restrictive constraint or a negative value for a more relaxed constraint. For example,

```
pt_shell> set_context_margin -type data -max 1.5
```

The *write_context* command adds 1.5 to the delays specified in *set_input_delay -max* and *set_output_delay -max* commands to make the maximum delay constraint more restrictive.

```
pt_shell> set_context_margin -type data -min 1.2
```

The *write_context* command subtracts 1.2 from the delays specified in *set_input_delay -min* and *set_output_delay -min* commands to make the minimum delay constraint more restrictive.

Adjustment as a Percentage of the Actual Delay

To adjust the delays by a specified percentage of the actual delays, use the *set_context_margin* command with the *-percent* option and specify the percentage using the *value* argument. For example,

```
pt_shell> set_context_margin -type data -max -percent 10
```

The *write_context* command adds 10 percent to the delays specified in *set_input_delay -max* and *set_output_delay -max* commands to make the maximum delay constraint more restrictive. For example, an actual maximum delay of 4.0 is changed to 4.4.

Adjustment as a Percentage of the Pin Slack

To adjust the delays by a specified percentage of the worst slack through each block input pin and output pin, use the *set_context_margin* command with the *-allocation pin_slack* and *-percent* options and specify the percentage using the *value* argument. For example,

```
pt_shell> set_context_margin -max -type data -allocation pin_slack
 -percent 40
```

The command allocates 40 percent of the maximum-delay slack to the block level and the remaining 60 percent to the top level. In each *set_input_delay -max* and *set_output_delay -max* command written by the *write_context* command, the delay value is determined as follows:

max_delay_constraint = actual_delay + (0.60 * slack_value)

For a positive slack, the constraint is tightened. For a negative slack, the constraint is relaxed.

The *-allocation pin_slack* option can be used only with the *-type data* option.

Here is a minimum delay constraint example:

```
pt_shell> set_context_margin -min -type data -allocation pin_slack
 -percent 45
```

min_delay_constraint = actual_delay - (0.55 * slack_value)

A pin slack allocation of 100 percent allocates all of the slack to the block and none to the top. This uses the actual top-level delay, the same as not setting any context margin.

A pin slack allocation of zero percent allocates none of the slack to the block and all to the top. This adds the full slack to the actual delay for a maximum delay constraint, or subtracts it from the actual delay for a minimum delay constraint.

Total Delay as a Percentage of the Clock Period

To specify the context delay as a percentage of the clock period, irrespective of the actual top-level delay, use the *set_context_margin* command with the *-allocation clock_period* and *-percent* options and specify the percentage using the *value* argument. For example,

```
pt_shell> set_context_margin -type data -max -allocation clock_period
 -percent 45
```

In each *set_input_delay -max* and *set_output_delay -max* command written by the *write_context* command, the delay value is 45 percent of the applicable clock period.

Absolute Specification (Override) in Time Units

To specify the context delay as an absolute number of time units, irrespective of the actual top-level delay, use the *set_context_margin* command with the *-allocation user_override* option and specify the time value using the *value* argument. For example,

```
pt_shell> set_context_margin -type data -max \\
  -allocation user_override 3.7 [get_pins blkB/in1]
```

In the *set_input_delay -max* command written by the *write_context* command for pin blkB/in1, the input delay value is 3.7 time units.

MIM Blocks

If you set the same type of margin on the same object more than once, the latest setting overrides the older setting. Blocks that belong to a multiply instantiated module (MIM) group are considered a single object in this respect. For example, suppose that blocks blkA and blkB belong to the same MIM group and you run the following commands:

```
set_context_margin -min -rise 0.5 [get_pins blkA/in1]
set_context_margin -min -fall 1.0 [get_pins blkB/in1]
```

Each command applies to all blocks in the MIM group, so the later margin setting replaces the older one for both blocks.

Examples

The following command adds a margin of 0.5 to all input and output maximum delay values, and subtracts that same value from all input and output minimum delay values, in the block context written out as *set_input_delay* and *set_output_delay* commands for data pins and *set_clock_latency -source* for clock pins. This applies to all blocks for which the tool is generating and writing out the block context.

```
pt_shell> set_context_margin 0.5
```

Use the *set_context_margin* command before *update_timing*. Otherwise, the *set_context_margin* command incurs the cost of another timing update.

The following command increases the calculated delays by 10 percent for the context generated and written out for block instance I2.

```
pt_shell> set_context_margin -max -delay -percent 10.0 -objects I2
```

The following script demonstrates usage of the *-allocation* option. Note that a more specific command has higher priority than a more general command.

```
# Set the delay constraint as a percentage of the launch clock period
create_clock -period 10 CLK
set_context_margin -type data -allocation clock_period -percent 50
# Input and output delays are set to 10*.50 = 5.0
#  irrespective of actual delays to input pins and from output pins

# Apply the context margin to a specific pin
set_context_margin -type data -allocation clock_period -percent 30
 [get_pins blkA/in1]
# Input delay for pin blkA/in1 is set to 10*.30 = 3.0

# Override the constraint setting for a specific pin
set_context_margin -type data -allocation user_override 1.23 [get_pins
 blkB/in1]
# Input delay for pin blkB/in1 is set to 1.23

# Allocate 60 percent of the slack to the block and the rest to the top
set_context_margin -type data -allocation pin_slack -percent 60 [get_pins
 blkB/in2]
# If worst setup slack is +2.0 and actual worst arrival at 3.5,
#  input delay for pin blkB/in2 is set to 3.5 + 2.0*(1.00-0.60) = 4.3

# Override the calculated source latency for a clock pin
set_context_margin -type clock -allocation user_override 4.5 [get_pins
 blkB/clk]
# Source latency of blkB/clk pin is set to 4.5
```

Feedback

See Also

- characterize_context
- report_context
- set_clock_latency
- set_input_delay
- set_output_delay
- write_context
- write_hier_data

---

set_coupling_separation

Creates a separation constraint on nets.

Syntax

```
status set_coupling_separation
   [-pairwise pnets]
   nets
```

Data Types

```
pnets                list
nets                 list
```

Arguments

`-pairwise pnets`

Specifies that coupling capacitances between only *pnets* and *nets* are excluded. This is referred to as "pairwise exclusion.

`nets`

Specifies the nets for which the separation constraint is applied. If you do not use the *-pairwise* option, the separation constraint is applied for this net with respect to all its aggressors.

Description

By default, all nets with coupling capacitors (non-filtered) are included in crosstalk and noise analysis as both victim and aggressor. This command creates a separation constraint on nets which means that all coupling capacitors on each net in *nets* are excluded for noise and crosstalk analysis.

Feedback

From an analysis point of view, this command has the same effect of applying both the *set_si_delay_analysis* and *set_si_noise_analysis* commands with the *-exclude* option and specifying *nets* as both victims and aggressors. However, this constraint takes precedence over the constraints set by the *set_si_delay_analysis* and *set_si_noise_analysis* command.

Instances of this command are recorded for output by the *write_changes* command. This feature allows you to communicate the separation constraint to other implementation tools along with netlist changes.

To remove the result of this command, use the *remove_coupling_separation* command.

To view the result of this command, use the *report_si_delay_analysis* or *report_si_noise_analysis* command with the *-coupling_separated* option.

Examples

The following example sets a separation constraint on all nets referred to by *CLK_NET_**.

```
pt_shell> set_coupling_separation [get_nets CLK_NET*]
1
```

The following example sets a separation constraint between net n3 and n1 and n2. It does not affect analysis of cross-coupling between n1 and n2 nor n3 and any other net.

```
pt_shell> set_coupling_separation -pairwise [get_nets {n1 n2}] [get_nets
 n3]
1
```

See Also

- remove_coupling_separation
- report_si_delay_analysis
- report_si_noise_analysis
- set_si_delay_analysis
- set_si_noise_analysis
- write_changes

---

set_create_lib_options

Specify options for create_lib command during synthesis

Syntax

int *set_create_lib_options*

Feedback

[-scale_factor *precision*]

Data Types

precision int

Arguments

-scale_factor *precision*

Specifies the length precision for the library. The length precision for the library and all of its reference libraries must be identical. The value is specified in terms of units per micron. By default, a length precision of 10000 is used, which implies one internal unit is equal to one Angstrom or 0.1 nm

Description

The *set_create_lib_options* command pass options for create_lib to synthesis module.

Examples

The following example shows how to specify create_lib options

```
prompt> set_create_lib_options -scale_factor 1
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_output_dir
- set_rtl_architect_shell
- set_presynth_options
- set_synth_options
- set_postsynth_options

---

set_cross_voltage_domain_analysis_guardband

Sets incremental derating factors to be used during cross-voltage-domain analysis for specified UPF supply nets or library cells.

Syntax

status *set_cross_voltage_domain_analysis_guardband*

```
-early | -late
-cell_check
derate_value
object_list
```

Data Types

derate_value	float
object_list	list

Arguments

`-early`

Applies the *derate_value* to early delays (shortest paths) during cross-voltage-domain analysis.

`-late`

Applies the *derate_value* to late delays (longest paths) during cross-voltage-domain analysis. The *-early* and *-late* options are mutually exclusive.

`-cell_check`

Applies the *derate_value* to cell_check arcs as with set_timing_derate the value specified combined with -late is used for setup checks and the value specified combined with -early is used for hold checks. This option is intended to derate the lauch capture delay difference inside of extracted timing models. To achieve this in pessimistic fashion it apllies the guardband derate by simple multiplication even to negative check values.

derate_value

Specifies the timing derate value applied as a scalar multiplier to cell delays during cross-voltage-domain analysis. This factor is applied in addition to any derate value specified with the *set_timing_derate* command or any advanced on-chip variation analysis invoked with the *timing_aocvm_enable_analysis* variable.

object_list

Specifies a collection of either UPF supply nets or library cells. If one or more supply nets are specified, the derating adjustment applies to all cells connected to those supply nets. If one or more library cells are specified, the derating adjustment applies to all instances of those cells. If the *object_list* argument is not used, the guard band setting applies to all UPF supply nets in the design.

Description

Cross-voltage-domain analysis uses two separate means of modeling the timing behavior of cells due to different supply voltage values. The preferred approach is to use the PrimeTime multivoltage delay calculation capability by providing libraries

characterized at the relevant corner voltages using the *define_scaling_lib_group* command. If library characterization information at different voltage values is not available, the *set_cross_voltage_domain_analysis_guardband* command can be used to model the change in timing behavior of cells at different voltage values by derating the calculated delays.

If no object list is provided in the command, the specified *derate_value* applies to all UPF supply nets in the design. If an object list is provided, it can be heterogeneous collection of either UPF supply nets or library cells.

Derating adjustment values can be set for both a UPF supply net collection and a library cell collection. Both adjustments are applied where applicable, one in addition to the other. Any new guard band set for a supply net collection overwrites any previous settings for supply nets. Similarly, any new guard band set for a library cell collection overwrites any previous settings for library cells.

The *set_cross_voltage_domain_analysis_guardband* derate is an incremental derate applied in addition to any other derate specified through the *set_timing_derate* command or advanced on-chip variation (AOCV) analysis. It is not recommended to use *set_cross_voltage_domain_analysis_guardband* in addition to *define_scaling_lib_group*. However, if you use both commands, the timing value obtained from the scaling lib group analysis is also derated according to the *derate_value* specified by the *set_cross_voltage_domain_analysis_guardband* command.

Examples

The following example sets a 10% delay derating for the late arrivals of on the supply nets VDD and VDD1:

```
pt_shell> set_cross_voltage_domain_analysis_guardband -late 0.1 \\
           [get_supply_nets {VDD VDD1]
```

When the *timing_enable_cross_voltage_domain_analysis* variable is set to *true*, cross-voltage-domain analysis is performed; the late delays of all cells supplied by the supply nets VDD and VDD1 are multiplied by 1.1. If ordinary derating is also specified with the *set_timing_derate* command, then both derating factors are applied to the calculated delays.

See Also

- set_timing_derate
- timing_enable_cross_voltage_domain_analysis

Feedback

set_current_command_mode

Syntax

string *set_current_command_mode*

-mode *command_mode* | -command *command*

```
string command_mode
string command
```

Arguments

`-mode command_mode`

Specifies the name of the new command mode to be made current. *-mode* and *-command* are mutually exclusive; you must specify one, but not both.

`-command command`

Specifies the name of the command whose associated command mode is to be made current. *-mode* and *-command* are mutually exclusive; you must specify one, but not both.

Description

The *set_current_command_mode* sets the current command mode. If there is a current mode in effect, the current mode is first canceled, causing the associated clean up to be executed. The initialization for the new command mode is then executed as it is made current.

If the name of the given command mode is empty, the current command mode is cancelled and no new command mode is made current.

A current command mode stays in effect until it is displaced by a new command mode or until it is cleared by an empty command mode.

If the command completes successfully, the new current command mode name is returned. On failure, the command returns the previous command mode name which will remain current and prints an error message unless error messages are suppressed.

Examples

The following example sets the current command mode to a mode called "mode_1"

```
shell> set_current_command_mode -mode mode_1
```

The following example clears the current command mode without setting a new mode.

```
shell> set_current_command_mode -mode ""
```

The following example sets the current command mode to the mode associated with the command "modal_command"

```
shell> set_current_command_mode -command modal_command
```

---

set_current_power_domain

Sets the specific power domains defined by the UPF *create_power_domain* command to be included in the power analysis.

Syntax

status *set_current_power_domain*

```
        list_of_power_domains
```

Data Types

list_of_power_domains	list

Arguments

list_of_power_domains

Specifies the power domains defined in the design to be included in the power analysis. If *list_of_power_domains* specifies an undefined power domain, the tool issues an error and does not change the current power domain setting.

Description

This command provides the control of calculating power consumption for the domains of interest. Domain-based power consumption data can be generated for a design with multiple power domains. Only the power dissipated in the blocks covered by the current power domain list is calculated and reported.

If you do not use this command, by default, all defined power domains are included in the power analysis.

This command has no effect on power results in non-UPF mode.

For both dynamic (internal and switching) and static (leakage) power, the domain-based power numbers are calculated based on which domain a cell belongs to. Before using this command, user has to define power domains by using the *create_power_domain* UPF command. The *get_current_power_domain* command returns the power domains being included in the power analysis.

The following power analysis results are affected by this command in UPF mode:

- Average power report • Time-based power report • Power-related attributes:
  - total_power
  - dynamic_power
  - internal_power
  - switching_power
  - leakage_power
  - intrinsic_leakage_power
  - gate_leakage_power
  - x_transition_power
  - glitch_power
  - peak_power
  - peak_time

To revert to the default behavior, which includes the whole design covered by all the defined power domains, use the *set_current_power_domain [get_power_domain *]* command.

Examples

The following example generates domain-based power analysis results. The design has two subblocks, InstA and InstB, which are covered by the PD1 and PD2 power domains, respectively. The first *report_power* generates the power analysis results for the whole design, which is the default behavior. The second *report_power* generates the power analysis results for the power consumed in the PD1 power domain. The third *report_power* generates the results for the power consumed in the PD2 power domain.

```
pt_shell> ...
  pt_shell> create_power_domain PD1 -elements {InstA}
  pt_shell> create_power_domain PD2 -elements {InstB}
  pt_shell> ...
  pt_shell> report_power
  pt_shell> set_current_power_domain PD1
  pt_shell> report_power
  pt_shell> get_current_power_domain
  pt_shell> set_current_power_domain PD2
  pt_shell> report_power
  pt_shell> get_current_power_domain
```

Feedback

See Also

- create_power_domain
- get_current_power_domain
- get_power_domains
- report_power
- report_power_domain

---

set_current_power_net

When the variable power_enable_multi_rail_analysis is true, specifes the power nets defined by UPF create_supply_net for power reporting. When false, sets the specific power nets to be included in the power analysis. By default (if not specified), the whole design covered by all the defined supply nets are included in the power analysis and report.

This command has no effect on power results in non-UPF mode.

Syntax

status *set_current_power_net*

```
        list_of_power_nets
```

Data Types

list_of_power_nets list

Arguments

list_of_power_nets

Specifies the power nets defined in the design to be included in the power analysis. If the *list_of_power_nets* option specifies an undefined supply net, an error is issued, and the current power net setting remains unchanged. All supply nets are selected as current_power_nets if the "all" is provided as list of power nets.

Description

The command behavior depends on the power_enable_multi_rail_analysis variable. When false, (the default); concurrent multi-rail analysis is disabled; and power for all the power nets in the design are calculated. This command controls which supply nets are to be included in the power analysis. When specified, only the power dissipation on the listed

supply nets is calculated and reported. To report power consumption per supply net, the command must be applied for each supply net, and the power reanalyzed per supply net.

When the power_enable_multi_rail_analysis variable is true, power per supply net is calculated, and the command controls the power reporting and attribute values. When specified, the power dissipation on the listed supply nets is accessible via the get_attribute and report_attribute commands, as well as the report_power command. To access the power attribute values per supply net, the command must be applied for each supply net, before issueing the get_attribute or report_attribute commands.

For both dynamic (internal and switching) and static (leakage) power, the power net-based power numbers are calculated based on the PG connection of the cell.

If using the CCS power model in the logic library:

- Dynamic_currents are based on pg_pin association.
- Leakage_currents are based on pg_pin association.
- Gate leakages are based on the related_power_pin for the input related pins.
- Switching powers are based on the related_power_pin for the output pins.

If using the NLPM Power model in the technology library with PG pin definitions:

- Internal powers are based on related_pg_pin attribute.
- Leakage powers are based on related_pg_pin attribute.
- Switching powers are based on the related_power_pin for the output pins.

Before using this command, you must define power nets by using the *create_supply_net* UPF command. You can use the *report_supply_net* command to show the defined power nets. You can use the *report_pg_pin_info* and *report_power_network* commands to show the power connections. the *get_current_power_net* command returns the supply nets being included in the power analysis.

The following power analysis results are affected by this command in UPF mode:

- Average power report • Time-based power report • Power-related attributes:
  - total_power
  - dynamic_power
  - internal_power
  - switching_power
  - leakage_power
  - intrinsic_leakage_power

- gate_leakage_power
- x_transition_power
- glitch_power
- peak_power
- peak_time

To revert to the default behavior, which includes the whole design covered by all the defined supply nets, use the *set_current_power_domain [get_supply_net *]* command.

Peak_power and peak_time per rail are only accessible when power_enable_multi_rail_analysis is false. When concurrent multi-rail analysis is performed, only the peak power and peak time for the all the supply nets is accessible. Per rail the peak power attributes are available only when concurrent multi-rail analysis is disabled.

Examples

The following example generates power net-based power analysis results. The design has two subblocks, InstA and InstB, which are covered by tje PD1 and PD2 power domains, respectively. There are six supply nets defined. Among them, four are power nets, and two are ground nets. The primary power net for power domain PD1 is VDDIS, and the primary power net for power domain PD2 is VDDGS.

The first *report_power* command generates the power analysis results for the whole design, which is the default behavior.

The second *report_power* command generates the power consumption associated with power net "VDDI".

The third *report_power* command generates the power consumption associated with power net VDDIS, which is the output of power switch top_header.

The fourth and fifth reports are for VDDG and VDDGS, respectively.

```
pt_shell> ...
pt_shell> create_power_domain PD1 -elements {InstA}
pt_shell> create_power_domain PD2 -elements {InstB}
pt_shell> create_supply_net VDDI  -domain PD1
pt_shell> create_supply_net VDDIS -domain PD1
pt_shell> create_supply_net VSS   -domain PD1
pt_shell> set_domain_supply_net PD1 -primary_power_net VDDIS
 -primary_ground_net VSS
pt_shell> connect_supply_net -ports InstA/LS_u1/VDDL VDDI
pt_shell> connect_supply_net -ports InstA/LS_u1/VDD VDDIS
pt_shell> connect_supply_net -ports InstA/LS_u1/VSS VSS
pt_shell> create_power_switch top_header \\
                     -domain PD1 \\
```

```
                      -output_supply_port {NSLEEPOUT VDDIS} \\
                      -input_supply_port {NSLEEPIN VDDI} \\
                      -on_state {state1 NSLEEPIN {CNTL}} \\
                      -control_port { CNTL ctrl }
pt_shell> ...
pt_shell> create_supply_net VDDG  -domain PD2
pt_shell> create_supply_net VDDGS -domain PD2
pt_shell> create_supply_net VSS   -domain PD2 -reuse
pt_shell> set_domain_supply_net PD2 -primary_power_net VDDGS
 -primary_ground_net VSS
pt_shell> create_power_switch gprs_header \\
                      -domain PD2 \\
                      -output_supply_port {NSLEEPOUT VDDGS} \\
                      -input_supply_port {NSLEEPIN VDDG} \\
                      -on_state {state1 NSLEEPIN {CNTL}} \\
                      -control_port { CNTL ctrl }
pt_shell> ...
pt_shell> report_power
pt_shell> set_current_power_net { VDDI }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDIS }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDG }
pt_shell> report_power
pt_shell> get_current_power_net
pt_shell> set_current_power_net { VDDGS }
pt_shell> report_power
pt_shell> get_current_power_net
```

See Also

- get_current_power_net
- set_current_power_domain
- get_current_power_domain
- create_supply_net
- get_supply_nets
- report_supply_net
- report_power_network
- report_power

Feedback

set_data_check

Defines a setup or hold timing check between two data signals arriving at specified points in the design.

Syntax

string *set_data_check*

```
-from from_object
   | -rise_from from_object
   | -fall_from from_object
-to to_object
   | -rise_to to_object
   | -fall_to to_object
[-setup]
[-hold]
[-clock clock_object]
[check_value]
```

Data Types

check_value	float
clock_object	collection
from_object	collection
to_object	collection

Arguments

`-from from_object`

Specifies the related pin or port of the data-to-data check. Both rising and falling transitions at this object are checked.

`-rise_from from_object`

Similar to the *-from* option, but only rising transitions at the pin are checked.

`-fall_from from_object`

Similar to the *-from* option, but only falling transitions at the pin are checked.

You must specify *-from*, *-rise_from*, or *-fall_from*.

`-to to_object`

Specifies the constrained pin or port of the data-to-data check. Both rising and falling transitions at this object are checked.

`-rise_to to_object`

Similar to the *-to* option, but only rising transitions at the pin are checked.

`-fall_to` *`to_object`*

Similar to the *-to* option, but only falling transitions at the pin are checked.

You must specify *-to*, *-rise_to*, or *-fall_to*.

For a setup check, the data must arrive at the constrained pin before the transition at the related pin by at least the *check_value* amount of time.

For a hold check, the data must remain valid at the constrained pin for at least the *check_value* amount of time after the transition at the related pin.

`-setup`

Performs only setup analysis in the data check.

`-hold`

Performs only hold analysis in the data check.

By default, the data check performs both setup and hold analysis using the same *check_value* for both. To specify different setup and hold times for the same data-to-data check, use two separate *set_data_check* commands.

`-clock` *`clock_object`*

Specifies the name of the single clock used for the data check. Use this option only if your design has multiple clocks per register and you want to select a single clock for the check.

`-sms_scenarios`

Limits the specification to only the provided SMS scenarios. When this option is not given, the specified value applies to all scenarios. The *get_sms_scenarios* command is used to specify the SMS scenario of interest.

`check_value`

Specifies the setup time or hold time requirement for the check.

Description

The *set_data_check* command specifies a data-to-data check performed between a "from" (related) object and a "to" (constrained) object. The "from" and "to" objects are ports or pins in the design where data signals arrive.

For a setup check, the data must arrive at the constrained pin before the transition at the related pin by at least the *check_value* amount of time.

For a hold check, the data must remain valid at the constrained pin for at least the *check_value* amount of time after the transition at the related pin.

The timing relationship between the contrained and related pins is based on zero cycles. This is different from usual sequential timing checks, which assume one cycle of delay from launch to capture by default.

The data check is treated as nonsequential; that is, the path goes through the related and constrained pins and is not broken at these pins.

To perform the data check, use the *report_timing* command. To report the specific data-to-data check, use the *report_timing* command with the *-to* option and specify the constrained pin as the "to" object.

The data signals arriving at the constrained pin could come from startpoints with multiple clocks in either of these situations:

- A single latch has multiple clocks propagating to the clock or gate pin.
- Different latches have different clocks propagating to their clock or gate pins. For such cases, these clock relations are checked separately. Similarly, data signals arriving at the related pin could also come from startpoints with multiple clocks. Use the *-clock* option to select a specific clock at the related pin for the analysis. Otherwise, the command analyzes multiple clocks and groups them into their respective clock groups.

A data-to-data check must have valid arrival times on both the constrained pin and the related pin. If an unclocked flip-flop does not launch any data, the related pin for the data-to-data check has no arrival time, which means that the path is still unconstrained.

Data-to-data checks are expected to operate on two data signals, and are not intended to perform clock skew checks. The tool does not prevent you from defining a data-to-data checks in the clock network, but the results of such checks might be incorrect.

To remove information set by the *set_data_check* command, use the *remove_data_check* command. *remove_data_check* can be used for per scenario timing check removal by adding the -sms_scenarios option.

Examples

The following command creates a data-to-data check from and1/B to and1/A with respect to the rising edge of signal at and1/B, using a setup and hold time of 0.4.

```
pt_shell> set_data_check -rise_from and1/B -to and1/A 0.4
```

The following command creates a data-to-data check from and1/B to and1/A with respect to the rising edge of signal at and1/B, coming from the clock domain of CK1, constraining only the falling edge of signal at and1/A, using a setup time of 0.5 and no hold check.

```
pt_shell> set_data_check -rise_from and1/B -fall_to and1/A -setup \\
            -clock [get_clock CK1] 0.5
```

The following commands specify different setup and hold values for the same data-to-data check.

Feedback

```
pt_shell> set_data_check -rise_from D2 -to D1 -setup 3.5
pt_shell> set_data_check -rise_from D2 -to D1 -hold 6.0
```

The following commands set a setup and hold timing check values between the same data signals, for different SMS scenarios using the -sms_scenarios option.

```
pt_shell> set_data_check -from D2 -to D1 -setup -sms_scenarios $low_low
 2.5
pt_shell> set_data_check -from D2 -to D1 -setup -sms_scenarios $high_low
 6.5
```

The following command removes the check value for a specific scenario.

```
pt_shell> remove_data_check -from D2 -to D1 -sms_scenarios $low_low
```

See Also

- remove_data_check
- report_timing
- update_timing

---

set_design_attributes

Sets the specified attributes and their value on specified cells.

Syntax

string *set_design_attributes*

```
[-elements list]
[-attribute name_value_pair]
[-models model_list]
[-is_soft_macro TRUE | FALSE]
[-is_hard_macro TRUE | FALSE]
[-switch_cell_type coarse_grain | fine_grain]
```

Data Types

list	list
name_value_pair	string
model_list	list
is_soft_macro	boolean
is_hard_macro	boolean
switch_cell_type	string

Feedback

Arguments

`-elements` *list*

Specifies the collection of cells on which the attributes must be set. Wildcards are accepted in cell names.

`-attribute` *name_value_pair*

Specifies the name and value of the attribute to be set on the cells specified by the *-elements* option.

`-models` *model_list*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-is_soft_macro` *TRUE* | *FALSE*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-is_hard_macro` *TRUE* | *FALSE*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-switch_cell_type` *coarse_grain* | *fine_grain*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignores this option.

`-lower_domain_boundary`

Defines the logical boundary of the root cells of the power domain in the hierarchical flow. PrimeTime reads and ignores this option.

Description

This command sets the attributes and their value on a list of cells specified by the *-elements* option.

Attributes can be set multiple times on the same cells. The latest value prevails.

PrimeTime supports a limited number of attributes. For attributes that are unsupported, PrimeTime ignores them and issues a warning message.

Examples

The following example sets the *merge_domain* attribute to *true* on the mid1 cell.

```
prompt> set_design_attributes -elements {mid1} -attribute merge_domain
 TRUE
1
```

Feedback

See Also

- report_attribute
- set_port_attributes

---

set_design_top

Sets or gets the current design in PrimeTime. It is a synonym for the *current_design* command.

Syntax

status *set_design_top*

`[design_name]`

Data Types

`design_name` `string`

Arguments

`design_name`

Specifies the working or focal design for many PrimeTime commands. If the *design_name* option is not specified, the *current_design* command returns a collection containing the current design. If the *design_name* option refers to a design that cannot be located, an error is issued and the working design remains unchanged.

Description

The UPF specification defines the *set_design_top* command that is in PrimeTime - implemented as a synonym to the *current_design* command.

Note: The *current_design* command removes all assertions (such as SDC, SPEF, and UPF). Therefore, the *set_design_top* command cannot be arbitrarily used in UPF scripts. It has to be used as the first command in UPF script, which is also used before any SDC or other assertions.

See Also

- current_design

---

set_device_parameter_margin

Set device physical parameter margin.

Feedback

Syntax

string *set_device_parameter_margin*

```
[-min min_value]
[-max max_value]
-parameter_name parameter_name
[-cell_check]
[-cell_delay]
[-increment]
[-override]
[object_list]
```

Data Types

min_value	float
max_value	float
i*parameter_name*	string
object_list	collection

Arguments

`-min` *min_value*

Specifies the value of device parameter margin for min analysis.

`-max` *max_value*

Specifies the value of device parameter margin for max analysis.

`-parameter_name` *parameter_name*

Specifies the physical parameter name to set margin.

`-cell_check`

Specify parameter value for cell timing checks only.

`-cell_delay`

Specify parameter value for cell delays only.

`-increment`

Specify an incremental value.

`-override`

Specify the margin is for override if parameter value is already defined in CTPM or PVT explorer condition. Without the option, default behavior is addition.

object_list

Specifies the cells or lib_cells to set device parameter margin. Without the option, the margin applies to whole design.

Feedback

Description

The *set_device_parameter_margin* command sets the device physical parameter margin on a group of cells or lib_cells, which allows you to add additional margin for physical parameter available in sensitivity augmented side-file defined by *define_sensitivity_lib_mapping*. Timing analysis with device parameter margin will refect the timing impact from phsyical parameter shift, and different margin can be applied for min and max analysis.

Examples

The following script enables Project Sicily Spice2Design flow, defines mapping between base library and sensitivity side-file, then set margin for SVT NMOS vth control parameter "delvtrandn_mos_svt" to 0.01 (min analysis of cell delays), 0.02 (max analysis of cell delays), 0.03 (min analysis of constraint arcs), and 0.04 (max analysis of constraint arcs).

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_spice2design_analysis true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]
set_device_parameter_margin -min 0.01 -parameter_name delvtrandn_mos_svt
 -cell_delay
set_device_parameter_margin -max 0.02 -parameter_name delvtrandn_mos_svt
 -cell_delay
set_device_parameter_margin -min 0.03 -parameter_name delvtrandn_mos_svt
 -cell_check
set_device_parameter_margin -max 0.04 -parameter_name delvtrandn_mos_svt
 -cell_check
```

See Also

- define_sensitivity_lib_mapping
- read_ctpm
- set_pvt_explorer_condition
- ps_enable_spice2design_analysis
- ps_enable_pvt_explorer_analysis

---

set_disable_auto_mux_clock_exclusivity

Disables automatic inference of MUXed clock exclusivity at specified cell output pins.

Syntax

```
status set_disable_auto_mux_clock_exclusivity
  [object_list]
```

Feedback

Data Types

object_list list

Arguments

object_list

Specifies a list of one or more MUX output pins for which automatic inference of MUXed clock exclusivity is disabled.

Description

Automatic inference of clock exclusivity at MUX output pins is enabled by setting the *timing_enable_auto_mux_clock_exclusivity* variable to *true*.

To prevent automatic inference of a specific MUX cell output as an exclusivity point, apply this command to the MUX output pin.

To undo the *set_disable_auto_mux_clock_exclusivity* command, use the *set_clock_exclusivity* command on the same output pin.

Examples

The following example prevents pins MUX32/Z and MUX33/Z from being automatically inferred as clock exclusivity points.

```
pt_shell> set_disable_auto_mux_clock_exclusivity {MUX32/Z MUX33/Z}
```

See Also

- remove_clock_exclusivity
- report_clock
- set_disable_auto_mux_clock_exclusivity

---

set_disable_check_timing

Sets certain objects to be excluded from check_timing signal_level and operating condition .

Syntax

boolean *set_disable_check_timinge*

```
[-check_type]
[-object_list]
```

Arguments

`-check_type`

Type of check_timing for which the timing checks needs to be disabled. Only two permissible values for this is operating_condition and signal_level.

`-object_list`

List of objects for which the timing check needs to be excluded.

Description

set_disable_check_timing command allows the users to set some objects to exclude them from timing checks in check_timing command. Currently exclude objects can be specified for operating_condition and signal_level checks using -check_type option.

For operating_conditions exclusion, only list of cells is allowed for the object_list. For signal_level exclusion, pins, lib_pins and ports are allowed as values to object_list.

See Also

- check_timing

---

set_disable_clock_gating_check

Disables the clock gating check for specified cell, pin, lib_cell, or lib_pin objects.

Syntax

string *set_disable_clock_gating_check*

object_list

Data Types

object_list list

Arguments

object_list

Specifies a list of cell, pin, lib_cell, or lib_pin objects for which the clock gating check is disabled.

Description

The *set_disable_clock_gating_check* command disables clock gating checks inferred automatically and possibly modified by the *set_clock_gating_check* command. It does not affect clock gating checks defined in the library for integrated clock-gating cells.

Feedback

You can disable clock gating checks on cells or pins in the current design or its subdesigns. Specify cells at lower levels in the design hierarchy as *instance1*/*instance2*/*cell_name*. Specify pins at lower levels in the design hierarchy as *instance1*/*instance2*/*cell_name*/*pin_name*. For subdesigns in the hierarchy, specify instance names, not design names.

Specifying a cell disables all gating checks in the cell. Specifying a pin disables any gating check in which the pin is the gating clock pin or gating enable pin. Specifying a lib_cell or lib_pin object affects all instances of that object used in the design.

To restore gating checks previously disabled by the *set_disable_clock_gating_check* command, use the *remove_disable_clock_gating_check* command. To globally disable all clock gating checks, including those defined in the library, set the *timing_disable_clock_gating_checks* variable to *true*.

Examples

The following command disables all clock gating checks on cell U123.

```
pt_shell> set_disable_clock_gating_check [get_cells U123]
```

The following command disables clock gating checks that use pin U123/I as a gating enable pin or gating clock pin.

```
pt_shell> set_disable_clock_gating_check [get_pins U123/I]
```

The following command disables clock gating checks for all instances of the mylib/AND1 library cell.

```
pt_shell> set_disable_clock_gating_check [get_lib_cells mylib/AND1]
```

The following command disables clock gating checks that use pin A1 as a gating enable pin or gating clock pin in instances of the mylib/AND1 library cell.

```
pt_shell> set_disable_clock_gating_check [get_lib_pins {mylib/AND1/A1}]
```

See Also

- remove_disable_clock_gating_check
- timing_disable_clock_gating_checks

---

set_disable_pg_pins

Disables PG pins of specific library cells, so they are not considered by timing analysis.

Syntax

status *set_disable_pg_pins*

Feedback

```
lib_cells
-pg_pins pg_pin_name_list
```

Data Types

```
lib_cells      collection
pg_pins        list
```

Arguments

lib_cells

Specifies the set of library cells on which PG pins will be disabled.

-pg_pins pg_pin_name_list

Specifies the names of the PG pins to be disabled.

Description

The *set_disable_pg_pins* command specifies the list of PG pins, belonging to specific library cells, to be disabled for timing analysis.

When a PG pin is disabled for a library cell, its connectivity and voltage value no longer affect the analysis.

This command can be used in both regular and scaling-group flows.

When this command is used in voltage-scaling flows, it must be specified before the scaling groups are defined by the *define_scaling_lib_group* command; it cannot be applied to a library after it is in a scaling group.

Note that the *define_scaling_lib_group* command has an *-excluded_rail_names* option that can be used to exclude a rail from an entire scaling group (all cells, not just specific cells).

Examples

The following script disables PG pin VDD on a libcell A in libray XYZ.

```
set_disable_pg_pin [get_lib_cell XYZ/A] -pg_pins {VDD}
```

See Also

- define_scaling_lib_group
- report_disable_pg_pins

---

set_disable_timing

Disables timing arcs in a circuit.

Feedback

Syntax

status *set_disable_timing*

```
[-from from_pin_name]
[-to to_pin_name]
object_list
comment
```

Data Types

```
from_pin_name     string
to_pin_name       string
object_list       list
comment           string
```

Arguments

`-from from_pin_name`

Specifies that arcs only from this pin on the specified cell are disabled. The *from_pin_name* value must be a valid library pin name corresponding to the cell in the *object_list* value. When used, the *-from* and *-to* options must be specified together. The *object_list* value must contain only cells, and the command specifies that arcs only between these two pins on the specified cells are disabled.

`-to to_pin_name`

Specifies that arcs only to this pin on the specified cell are disabled. The *to_pin_name* value must be a valid library pin name corresponding to the cell in the *object_list* value. When used, the *-from* and *-to* options must be specified together. The *object_list* value must contain only cells, and the command specifies that arcs only between these two pins on the specified cells are disabled.

object_list

Specifies a list of cells, pins, lib-cells, lib-pins, ports, or timing-arcs that are disabled. This list can include library objects.

comment

Associates a string description with the command for tracking purposes. You must begin and end the comment string with double quotation marks ("). Currently, this option is disabled as a default. To enable it set pt_enabled_comment_option gvar to true.

Description

Disables timing through the specified cells, pins, ports, or timing arcs in the *current_design* command.

Feedback

Any cell, pin, port, or timing arc in the *current_design* command or its subdesigns can be disabled. Cells at lower levels in the design hierarchy are specified as *instance1*/*instance2*/*cell_name*. Pins at lower levels in the design hierarchy are specified as *instance1*/*instance2*/*cell_name*/*pin_name*. Ports at lower levels in the design hierarchy are specified as *instance1*/*instance2*/*cell_name*/*port_name*. Timing arcs have to be collected using the *get_timing_arcs* command.

Note: For subdesigns in the hierarchy, you must specify instance names instead of design names.

When the timing through a cell is disabled, only those cell arcs are disabled that lead to the output pins of the cell. When the timing through a pin or port is disabled, only those cell arcs that lead from a load pin and to a driver pin are disabled. For bidirectional pins or ports the cell arcs from the load side and to the driver side are disabled.

When the *-from* and *-to* option pins are specified, all arcs between these two pins on the cell are disabled.

To view disabled timing arcs in the current design, use the *report_disable_timing* command. To restore timing arcs disabled by the *set_disable_timing* command, use the *remove_disable_timing* command. The *reset_design* command removes all user-specified attributes from the current design, including those set by the *set_disable_timing* command.

Examples

Consider the following reported path:

```
pt_shell> report_timing -input_pins

****************************************
Report : timing
         -path full
         -delay max
         -input_pins
         -max_paths 1
Design : counter
 ...
****************************************


  Startpoint: ffb (rising edge-triggered flip-flop)
  Endpoint: CO (output port)
  Path Group: (none)
  Path Type: max

  Point                                          Incr       Path
  -----------------------------------------------------------------
  ffb/CP (FD3)                                   0.00       0.00 r
  ffb/QN (FD3)                                   2.42       2.42 r
  m/C (AN4)                                      0.00       2.42 r
  m/Z (AN4)                                      1.12       3.54 r
```

```
  CO/A (AN2)                                0.00       3.54 r
  CO/Z (AN2)                                0.48       4.02 r
  CO (out)                                  0.00       4.02 r
  data arrival time                                    4.02
  ---------------------------------------------------------------
  (Path is unconstrained)
```

Assuming you want to disable timing arcs through the CO cell, you can view all timing arcs of the CO cell by examining its library cell, AN2.

```
pt_shell> report_lib -timing lsi_10k { AN2 }

****************************************
Report : library
        -timing_arcs
Library: ...
 ...
****************************************

Library Cells:

   Attributes:
    b - black box (function unknown)
    d - dont_touch
    s - state table
    u - dont_use
    A - abstracted timing model
    E - extracted timing model
    I - Interface timing spec model (ITS)
    S - Stamp timing model
    Q - Quick timing model (QTM)

                              Arc                    Arc Pins
   Lib Cell  Attributes     #  Type/Sense       From         To          When

 ------------------------------------------------------------------------------
   AN2                      0  positive_unate   A            Z
                            1  positive_unate   B            Z
```

To disable the timing arc used in the preceding reported path, use the *set_disable_timing* command:

```
pt_shell> set_disable_timing -from A -to Z [ get_cells { CO } ]

pt_shell> report_timing

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
 ...
****************************************
```

```
  Startpoint: ffa (rising edge-triggered flip-flop)
  Endpoint: QA (output port)
  Path Group: (none)
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  ffa/CP (FD2)                             0.00       0.00 r
  ffa/Q (FD2)                              1.42       1.42 f
  QA/Z (IV)                                0.38       1.80 r
  QA (out)                                 0.00       1.80 r
  data arrival time                                   1.80
  -----------------------------------------------------------
  (Path is unconstrained)
```

Alternatively, the same arc is disabled as follows:

```
pt_shell> set_disable_timing [get_timing_arcs -from { CO/A } -to
 { CO/Z } ]
```

See Also

- get_timing_arcs
- remove_disable_timing
- report_disable_timing
- report_lib
- report_timing
- reset_design
- timing_disable_cond_default_arcs

---

set_distributed_parameters

Configures the distributed environment.

Syntax

status *set_distributed_parameters*

```
[-collection_levels collection_level]
 [-script script]
 [-worker_common_user_data worker_script]
```

Feedback

Data Types

```
collection_level    string
script              string
worker_script       string
```

Arguments

`-script` *script*

The user-defined/custom wrapper script to be used when launching remote processes. If this option is not specified, the wrapper is taken by default to be $synopsys_root/bin/pt_shell.

`-collection_levels` *collection_level*

Specifies the number of collection levels to traverse when retrieving collection attributes from the workers. The higher this collection level is set, the more memory is potentially consumed at the manager.

`-worker_common_user_data` *worker_script*

Specifies the filename of the Primetime script to be sourced at the startup of each worker process. The script may contain definitions of common Tcl procedures, setting of common variables and so on. The script is run after sourcing .synopsys_pt.setup files and is not intented to replace .synopsys_pt.setup.

Description

The *set_distributed_parameters* command allows you to configure the distributed environment. The command should be issued before the *start_hosts* command for your configurations to take effect.

Examples

In the following example one worker process is allocated on a host called platinum1. The process is launched using the *start_hosts* command.

```
pt_shell> set_host_options -num_processes 1 platinum1
1
pt_shell> set_distributed_parameters -script /usr/pt_shell
1
pt_shell> start_hosts
1] Launched: rsh -n -l user platinum1 /usr/pt_shell
             -env_start ...
             ...
               Status: Forking successful
               Stdout: Process group is (4492)
               Stderr: **<<EMPTY>>**
```

Feedback

```
 ------------------------------------------------------------------------
---------------
  Distributed farm creation timeout : 21600 seconds
  Waiting for   1 (of   1) distributed hosts (Mon Dec  9 05:04:36 2013)
  Waiting for   0 (of   1) distributed hosts (Mon Dec  9 05:04:46 2013)

 ------------------------------------------------------------------------
---------------
```

In the following example, the *-collection_levels* option is used to restrict the amount of information returned by the *get_distributed_variable* command. By setting the collection level to *4* only four levels of collections can be traversed.

```
pt_shell> set_distributed_parameters -collection_level 4
pt_shell> remote_execute { set m_pins [get_pins A]}
pt_shell> get_distributed_variables m_pins -attr {clocks sources}
pt_shell> echo [set clock1 [get_attribute $m_pins(scen1) clocks]]
_sel12
pt_shell> echo [set clock_source1 [get_attribute $clock1 sources]]
_sel13
pt_shell> echo [set clock2 [get_attribute $clock_source1 clocks]]
_sel14
pt_shell> echo [set clock_source2 [get_attribute $clock2 sources]]
_sel15
pt_shell> echo [set clock3 [get_attribute $clock_source2 clocks]]
Warning: Attribute 'clocks' does not exist on port 'CLK' (ATTR-3)
```

In the example below, the *-worker_common_user_data* option specifies the script file to be sourced at the start of each worker process:

```
pt_shell> set_distributed_parameters
 -worker_common_user_data /home/user/set_worker_env.tcl
```

See Also

- set_host_options
- start_hosts

---

set_distributed_power_analysis_options

Sets the options for distributed power analysis in PrimePower. Currently supported only in time based analysis.

Syntax

integer *set_distributed_power_analysis_options*

```
[-partition_style string]
[-common_scripts string]
```

```
[-merge_output_waveform]
[-merge_activity_waveform]
[-keep_overlapped]
```

Data Types

string string

Arguments

-partition_style

This option specifies the scheme used for partitioning the task execution among different workers.

The schemes currently available are : partition_by_time (Default, if no scheme is specified) partition_by_design partition_by_design_and_time

If scheme specified is 'partition_by_time', the partitioning of task is done by dividing FSDB time interval.

If the scheme specified is 'partition_by_design', the design is partitioned into multiple sub designs and each worker runs a sub design.

If the scheme specified is 'partition_by_design_and_time', the design is partitioned into sub designs and if required, FSDB time based partitioning is also done. This is suitable for huge design sizes with virtual FSDB where processing entire design in a given machine may not be feasible.

-common_scripts

This option is required when it is required to do a common setting in all the workers. Eg - If 'set_switching_activity' is done in master, it should be put in a file, say 'common_script', and this file name should be passed as argument to '-common_scripts'

-merge_output_waveform

Merge output fsdb waveform generated inside partitions into single fsdb waveform.

-merge_activity_waveform

Merges the output activity waveforms generated inside partitions into single activity waveform.

-keep_overlapped

This can be used to smoothen the merged power waveform if it has holes at the borders of partition windows.

Feedback

Description

This command specifies the options for distributed power analysis. With distributed analysis, it is expeceted to see runtime improvement.

Examples

The following example will cause PrimePower to do distributed power analysis.

```
pt_shell> set power_enable_distributed_analysis true
pt_shell> set_distributed_power_analysis_options -partition_style
 partition_by_design_and_time -common_scripts common_script
pt_shell> update_power
```

See Also

- power_enable_concurrent_event_analysis
- set_host_options

---

set_distributed_variables

Passes Tcl variables from the manager process to the worker processes.

Syntax

status *set_distributed_variables*

```
[-pre_commands pre_command_string]
[-post_commands post_command_string]
object_list
```

Data Types

pre_command_string	string
post_command_string	string
object_list	list

Arguments

`-pre_commands` *pre_command_string*

Specifies a string of commands to be executed in the worker context before the setting of variables. To group commands, use the semicolon (;) character. The maximum size of a command is 1000 chars.

`-post_commands` *post_command_string*

Specifies a string of commands to be executed in the worker context after the setting of variables. To group commands, use the semicolon (;) character. The maximum size of a command is 1000 chars.

object_list

Specifies a list of objects.

Description

This command is available only if you invoke the pt_shell with the *-multi_scenario* option.

This command sets variables at the worker processes running PrimeTime scenarios given a set of variables at the manager. If the variable to be sent is a simple Tcl list, the Tcl list is recreated at workers for all scenarios in focus. If the variable is an array indexed by the scenario name, the variable is set for each scenario in focus that has an entry in the array.

Examples

The following example sets a scenario_id variable for each scenario and a max_fanout variable in both scenarios. In the scen1 scenario, the scenario_id variable is created and set to 1. In the scen2 scenario, the scenario_id variable is created and set to 2. In both scenarios, the max_fanout variable is created and set to 10.

```
pt_shell> set max_fanout 10
pt_shell> array set scenario_id {scen1 1 scen2 2}
pt_shell> set_distributed_variables {scenario_id max_fanout}
```

See Also

- get_distributed_variables

---

set_domain_supply_net

Sets the primary power net and primary ground net of an existing power domain.

Syntax

int *set_domain_supply_net*

```
domain_name
-primary_power_net supply_net_name
-primary_ground_net supply_net_name
```

Data Types

```
domain_name           string
supply_net_name       string
```

Arguments

domain_name

Specifies the name of the power domain to set.

If there is no such power domain with the specified name in the current scope, the command fails.

`-primary_power_net` *`supply_net_name`*

Specifies the power net of the power domain.

If there is no such supply net with the specified name in the current scope, the command fails. If the supply net is not associated with the specified power domain, the command also fails.

`-primary_ground_net` *`supply_net_name`*

Specifies the ground net of the power domain.

If there is no such supply net with the specified name in the current scope, the command fails. If the supply net is not associated with the specified power domain, but is associated with other power domains, the command also fails.

Description

The *set_domain_supply_net* command enables you to set the primary power net and the primary ground net of a power domain. All extent of the power domains shares the same power/ground supply until explicitly excluded. Here "explicitly excluded" means it is explicitly connected with another supply net (non primary power/ground supply net) by the following commands: *connect_supply_net*, *set_isolation*, *set_retention*. This command fails if the domain already has a primary power and ground net.

The supply net must be created in the same power domain first before it is set as the primary power/ground supply net. Use the *create_supply_net* command to create a supply net at a specified power domain.

The command returns 1 if is is successful, and returns 0 otherwise.

Examples

The following example creates two supply_net on power_domain PD1, then sets them as primary power/ground net.

```
prompt> create_power_domain PD1 -elements INST_1
PD1
prompt> create_supply_net A_VDD -domain PD1
A_VDD
prompt> create_supply_net A_VSS -domain PD1
A_VSS
prompt> set_domain_supply_net PD1 -primary_power_net A_VDD \
-primary_ground_net A_VSS
1
```

Feedback

See Also

- create_power_domain
- create_supply_net
- connect_supply_net

---

set_dont_override

Specifies the objects for which user-defined budgeted constraints should not be automatically overridden during the HyperScale analysis flow. This command is only effective during block-level analyses.

Syntax

string *set_dont_override*

```
[-include arc_delay | case_value | pin_transition]
object_list
```

Data Types

```
object_list     list
```

Arguments

`-include arc_delay | case_value | pin_transition`

Specifies the name of the include list during the block-level analysis which can ignore the data from the top level and the block level can use the data from itself. Currently you can use -include arc_delay, case_value or pin_transition during block-level analysis to ignore the arc_delay, case_value or pin_transition respectively from the top level and use the arc_delay, case_value or pin_transition from block itself.

object_list

Specifies the design or ports for which boundary budgeted constraints should be maintained without automatic override.

Description

This command is used to select the ports or the entire block for which the boundary timing context pushed down from parent level analysis should not be used to override user-defined budgets.

In the HyperScale analysis flow, a top-level run pushes accurate and up-to-date boundary timing and noise context for the subblocks represented by abstraction; the subsequent block-level run automatically loads the latest timing context for the block boundary and override user-defined I/O context to perform the block level analysis. These boundary

timing context for the input and output ports override user-defined I/O timing constraints, such as *set_input_delay*, *set_output_delay*, *set_input_transition*, *set_clock_latency*, *set_timing_derate* and *set_input_noise*.

In a design flow, sometimes you might want the block-level design to meet a fixed timing budget prescribed for the block, instead of allowing PrimeTime to automatically override with actual timing context, which can represent some intermediate state of the parent level or other surrounding blocks. Particularly in the early design flow, or for certain portion of the design with known and clear timing specification.

You can use this command only when you also use the *set_hier_config -parent* command, meaning that the current run is for a block, and *hier_enabled_analysis* is set to *true*.

Examples

Assuming a design with two levels of hierarchy, where TOP instantiates a subblock BOT, and the named scenario of interest is fast_func. To perform hierarchical timing analysis for BOT and automatically generate a reduced and accurate timing representation for its parent level analysis, use the following commands:

```
pt_shell> set hier_enable_analysis true
pt_shell> set_hier_config -parent -name "fast_func" -path /design/TOP
pt_shell> set_hier_config -name "fast_func" -path /design/blocks/BOT
```

In addition, for the block BOT, it is necessary to freeze the timing budget defined for the bused port DATAIN so that PrimeTime does not automatically use the actual context pushed from TOP to override your settings, use the following command:

```
pt_shell> set_dont_override [get_port DATAIN*]
```

See Also

- remove_dont_override
- link_design
- report_constraint
- save_session
- set_hier_config
- update_timing
- hier_enable_analysis

Feedback

set_dont_touch

Sets the *dont_touch* attribute on cells, nets, designs, or library cells to prevent those objects from being modified or replaced during ECO fixing.

Syntax

status *set_dont_touch*

```
object_list
[value]
```

Data Types

```
object_list    list
value          Boolean
```

Arguments

object_list

Specifies a list of cells, nets, designs, or library cells, preventing those objects from being modified or replaced during ECO fixing.

value

Sets the value, either *true* or *false*, for the *dont_touch* attribute of the specified objects. The default is *true*. Use the value *false* to cancel the effects a previous *set_dont_touch* command.

Description

This command sets the *dont_touch* attribute on cells or nets in the current design, or on designs or library cells, to prevent those objects from being modified or replaced during ECO fixing by the *fix_eco_drc*, *fix_eco_power*, and *fix_eco_timing* commands. The specified objects must be in one of the target libraries or in a library already loaded into memory.

When set on a cell instance, that cell cannot be sized, removed, or replaced. For a hierarchical cell, the *dont_touch* setting applies to all its lower-level cells, except where their *dont_touch* attribute is set to *false*.

When set on a net, buffer insertion is not allowed on that net. Driver and load cells connected to the net can still be modified or removed. For a net that crosses hierarchical boundaries, the *dont_touch* setting applies only to the named net segment and does not propagate across the hierarchical boundaries.

When set on a design, the *dont_touch* attribute applies to all instances of that design. When set on the *[current_design]*, the *dont_touch* attribute applies from the top level down. Lower-level hierarchical values take precedence over higher-level values, and leaf-level attributes take precedence over design-level attributes.

Feedback

When set on a library cell, the *dont_touch* setting applies to all instances of that cell. It affects only the library cells loaded into memory, not the library stored on disk.

You can query the *dont_touch* attribute by using the *get_attribute* command. To cancel the effects of a *set_dont_touch* command, run the command again and specify the *value* as *false*. A *false* setting on an object overrides a *true* setting on the parent object.

In distributed multi-scenario analysis (DMSA), you should apply the *set_dont_touch* command in all the scenarios.

The *dont_touch* attribute is characterized by the *characterize_context* command and exported to other tools by the *write_context* command.

When the *dont_touch* attribute on a design or library cell is written to scripts, the target objects vary based on the script format. A *dont_touch* design is applied to a design for PrimeTime and to a reference for Design Compiler. Similarly, a *dont_touch* library cell is applied to a lib_cell for PrimeTime, and to a reference for Design Compiler.

If your database contains a *dont_touch* attribute on an object, and you explicitly use the *set_dont_touch* command to set the value to *false* for that object, the *write_script* command does not write this information.

The *dont_touch* attribute is inherited by cells if the cell is an instance of a design or library cell that has the *dont_touch* attribute. In these cases, the *get_attribute* command returns *true* for the *dont_touch* attribute for that cell. This affects the result of the *get_attribute* command but does not actually transfer the *dont_touch* attribute to the cell. So the output of the *write_script* command does not contain extra *set_dont_touch* commands applied to cells.

Examples

The following commands specify not to modify the "block1" and "analog1" cells during ECO fixing, but allow the net "N1" to be modified.

```
pt_shell> set_dont_touch [get_cells {block1 analog1}]
1
pt_shell> set_dont_touch  [get_nets N1] false
1
```

In the following example, applying a *dont_touch* attribute to design d1 affects its instances. The *get_attribute* command returns *true* for the *dont_touch* attribute. The example also shows the difference in the *write_script* command output for the PrimeTime and Design Compiler tools; the applicable line of output for each tool is shown below.

```
pt_shell> get_attribute [get_cells u1] ref_name
d1

pt_shell> get_attribute [get_cells u1] dont_touch
false
```

```
pt_shell> set_dont_touch [get_designs d1]
1

pt_shell> get_attribute [get_cells u1] dont_touch
true

pt_shell> write_script -format ptsh

 Script contains this command:
 set_dont_touch [get_designs "design1.db:d1"]

pt_shell> write_script -format dcsh

 Script contains this command:
 set_dont_touch find(reference, "d1")
```

The following example shows that the *dont_touch* setting applied to a net at the top level, sdramA, is not propagated across a hierarchical boundary to a lower-level connected net, I_ORCA_TOP/sdA.

```
pt_shell> get_nets sdramA
{"sdramA"}
pt_shell> get_nets -segments sdramA
{"sdramA", "I_ORCA_TOP/sdA"}
pt_shell> get_attribute [get_nets -segments sdramA] dont_touch
Warning: Attribute 'dont_touch' does not exist on net 'sdramA' (ATTR-3)
Warning: Attribute 'dont_touch' does not exist on net
 'I_ORCA_TOP/sdA' (ATTR-3)

pt_shell> set_dont_touch [get_nets sdramA] true
1
pt_shell> get_attribute [get_nets -segments sdramA] dont_touch
Warning: Attribute 'dont_touch' does not exist on net
 'I_ORCA_TOP/sdA' (ATTR-3)
true
```

See Also

- fix_eco_drc
- fix_eco_power
- fix_eco_timing
- get_attribute
- set_dont_touch_network
- set_dont_use
- write_script

Feedback

set_dont_touch_network

Sets the dont_touch_network attribute on clocks, pins, or ports in the current design to prevent cells and nets in the transitive fanout of the set_dont_touch_network objects to prevent synthesis from replacing or modifying them during optimization.

Syntax

```
status set_dont_touch_network
  object_list
```

Data Types

```
object_list      list
```

Arguments

object_list

Specifies a list of clocks, pins, or ports.

Description

Sets the *dont_touch_network* attribute on clocks, pins, or ports in the current design. It also assigns *dont_touch* attributes to all cells and nets in the transitive fanout of the *dont_touch_network* objects so that they are not modified or replaced during optimization.

The *set_dont_touch_network* command is intended primarily for clock circuitry. Placing a *dont_touch_network* attribute on a clock object prevents synthesis from modifying the clock buffer network.

This attribute is characterized by the *characterize_context* command and exported to synthesis using the *write_context* command.

If you specify a pin or a port, all nets and cells in its transitive fanout are recursively marked *dont_touch*. This recursion propagates through combinational cells but stops at registers (without marking the registers as *dont_touch*). An element is recognized as a register only if it has setup or hold constraints. If you specify an input pin, its own cell is marked *dont_touch* but its connected net is not marked *dont_touch*. If you specify an output pin, its own cell is not marked *dont_touch* but its connected net is marked *dont_touch*.

Refer to the *set_dont_touch* command man page in Design Compiler for a complete description of the *dont_touch* attribute and its effect.

Use the *reset_design* command to remove the *dont_touch_network* and *dont_touch* attribute.

Examples

The following command adds a *dont_touch_network* attribute to the port named "clock_in".

```
pt_shell> set_dont_touch_network [get_ports clock_in]
```

The following command adds the *dont_touch* attribute to a net named net1 and a cell named cell2 but not to the cell named cell1:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y cell2/A}
prompt> set_dont_touch_network [get_pins cell1/Y]
1
```

The following command adds the *dont_touch* attribute to a net named net1 and cell named cell1 but not on the net named net0 and register reg1:

```
prompt> get_pins -of_objects [get_nets net1]
{cell1/Y reg1/D}
prompt> get_pins -of_objects [get_cells cell1]
{cell1/A cell1/Y}
prompt> get_nets -of_objects [get_pins cell1/A]
{net0}
prompt> set_dont_touch_network [get_pins cell1/A]
1
```

See Also

- characterize_context
- report_context
- set_dont_touch
- write_context

---

set_dont_use

Sets the *dont_use* attribute on library cells to exclude them from the target library during optimization.

Syntax

```
status set_dont_use
   object_list
   [true | false]
```

Data Types

```
object_list    list
```

Feedback

Arguments

object_list

Specifies a list of objects (library cells, modules, or implementations) on which the *dont_use* attribute is to be set.

`true | false`

Specifies whether the *dont_use* attribute is set to *true* (the default) or *false*.

Description

This command sets the *dont_use* attribute on specified library objects, so that they are not used for ECO. The specified objects must be in one of the target libraries or in a library already loaded into memory.

use this command with care, especially when changing the value of the *dont_use* attribute from *true* to *false*. Follow the design requirements that establish which library cells can be used for ECO. Disabling library cells can have an impact on PrimeTime ECO results depending on the choices of library cells available for ECO.

This command affects only the copy of the library that is currently loaded into memory and has no effect on the version that exists on disk.

In the distributed multi-scenario analysis flow, you should apply the *set_dont_use* command in all the scenarios so that the specified library cells are not used during optimization, such as when using the *fix_eco_timing* command.

Examples

Setting the *dont_use* attribute to *true* on a library cell disables it from ECO usage while setting it to *false* enables it for ECO usage.

The following command sets the *dont_use* attribute on the G1 and G2 lib_cells in the mylib library:

```
pt_shell> set_dont_use {mylib/G1 mylib/G2}
1
```

The following command unsets the *dont_use* attribute on the G1 lib_cell in the mylib library:

```
pt_shell> set_dont_use {mylib/G1} false
1
```

See Also

- fix_eco_drc
- fix_eco_power

Feedback

- fix_eco_timing
- get_attribute
- report_lib

---

set_drive

Sets the resistance to a specified value on specified input or inout ports in the current design.

Syntax

status *set_drive*

```
[-rise]
[-fall]
[-min]
[-max]
resistance_value
port_list
```

Data Types

resistance_value	float
port_list	list

Arguments

`-rise`

Specifies that the *resistance_value* argument is used to drive the ports only for the rising case.

`-fall`

Specifies that the *resistance_value* argument is used to drive the ports only for the falling case.

`-min`

Applies only to designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* argument is the minimum resistance.

`-max`

Applies only to designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* argument is the maximum resistance.

resistance_value

Specifies a nonnegative port drive resistance value for the ports in the *port_list*. The value must be >= 0; units must be the same as those in the logic library.

port_list

Specifies a list of input or inout ports in the current design, in which the *resistance_value* argument is set.

Description

This command sets the resistance to a specified value on specified input or inout ports in the current design. PrimeTime models the driver as a resistance and uses that value to calculate the wire delay of the port. To view the drive that is set, use the *report_port -drive* attribute.

Depending on the options used, the *set_drive* command sets these attributes on the specified ports:

```
drive_resistance_fall_max
drive_resistance_fall_min
drive_resistance_rise_max
drive_resistance_rise_min
```

If the *set_drive* command is issued without any options, the *drive_resistance_fall_max* and *drive_resistance_rise_max* attributes are set; in min-max mode, the other two attributes are also set.

If the *set_drive* command is issued with only the *-rise* option, only the *drive_resistance_rise_max* attribute is set; in min-max mode, the *drive_resistance_rise_min* attribute is also set.

If the *set_drive* command is issued with only the *-fall* option, only the *drive_resistance_fall_max* attribute is set; in min-max mode, the *drive_resistance_fall_min* attribute is also set.

Drive resistance is the output resistance of the cell that drives the port, so that a higher drive resistance means less drive capability and longer delays. Thus, a drive *resistance_value* of 0 is infinite drive, or no delay between the ports and all ports connected to them. Setting *resistance_value* to 0 is the same as removing the drive resistance with the *remove_drive_resistance* command.

There are two other methods of describing port drive capability. *set_driving_cell* command or the *set_input_transition* command. The most recent drive command has precedence.

Examples

The following example sets the drive resistance to 5 on all input ports in the current design, and displays the ports on which the drive resistance has been set.

```
pt_shell> set_drive 5 [all_inputs]
1
pt_shell> report_port -drive
****************************************
Report : port
```

Feedback

```
        -drive
Design : counter
*************************************

                 Resistance          Transition
Input Port     Rise      Fall      Rise      Fall
-------------------------------------------------------------
A              5.00      5.00      --        --
B              5.00      5.00      --        --
C              5.00      5.00      --        --
CL             5.00      5.00      --        --
CLK            5.00      5.00      --        --
D              5.00      5.00      --        --
JOKE           5.00      5.00      --        --
L              5.00      5.00      --        --
P              5.00      5.00      --        --
RESET          5.00      5.00      --        --
T              5.00      5.00      --        --
```

See Also

- remove_drive_resistance
- report_port
- set_load

---

set_drive_resistance

Sets drive resistance for input or inout ports.

Note: This command is obsolete, and has been replaced by the *set_drive* command. Use the *set_drive* command instead.

Syntax

string *set_drive_resistance*

```
[-rise]
[-fall]
[-min]
[-max]
resistance
port_list
```

Data Types

```
resistance            float
port_list             list
```

Feedback

Arguments

`-rise`

Specifies to use the resistance to drive the ports for the rising case.

`-fall`

Specifies to use the resistance to drive the ports for the falling case.

`-min`

Specifies to use the resistance to drive the ports for the minimum case.

`-max`

Specifies to use the resistance to drive the ports for the maximum case.

`resistance`

Specifies a nonnegative resistance value for the ports. Port drive resistance (value >= 0).

`port_list`

Specifies a list of input or inout port names of the current design on which the drive attributes are to be set.

Description

Model the driver as a resistance and use that value to calculate the wire delay of the port. The *report_port -drive* command can be used to view the drive that is set.

Drive resistance is the output resistance of the cell that drives the port; such that a higher drive resistance means less drive capability and longer delays. Thus, a drive *resistance* of 0 is infinite drive, or no delay between the ports and all connected to them. This is the same as removing the drive resistance with the *remove_drive_resistance* command.

Drive resistance must be in units with the technology library.

Examples

Here is an example of the *set_drive_resistance* command in use.

```
pt_shell> set_drive_resistance 5 [all_inputs]
1
pt_shell> report_port -drive
****************************************
Report : port
         -drive
Design : counter
****************************************

                 Resistance          Transition
Input Port     Rise      Fall      Rise      Fall
```

Feedback

```
------------------------------------------------------------
A              5.00      5.00      --        --
B              5.00      5.00      --        --
C              5.00      5.00      --        --
CL             5.00      5.00      --        --
CLK            5.00      5.00      --        --
D              5.00      5.00      --        --
JOKE           5.00      5.00      --        --
L              5.00      5.00      --        --
P              5.00      5.00      --        --
RESET          5.00      5.00      --        --
T              5.00      5.00      --        --
```

See Also

- remove_drive_resistance
- report_port
- set_drive
- set_load

---

set_driving_cell

Sets the port driving cell.

Syntax

string *set_driving_cell*

```
-lib_cell lib_cell_name
[-rise]
[-fall]
[-min]
[-max]
[-library lib_name]
[-pin pin_name]
[-from_pin from_pin_name]
[-no_design_rule]
[-input_transition_rise rtran]
[-input_transition_fall ftran]
[-clock clock_name]
[-clock_fall]
[-sms_scenarios sms_scenarios_list]
port_list
```

Data Types

```
clock_name          string
from_pin_name       string
ftran               float
```

```
lib_cell_name          string
lib_name               string
pin_name               string
port_list              list
rtran                  float
sms_scenarios_list     collection
```

Arguments

`-lib_cell` *lib_cell_name*

Specifies the name of the library cell used to drive the ports. You must use the *-pin* option if the cell has more than one output pin. If different cells are needed for the rising and the falling cases, use separate commands with the *-rise* or *-fall* option.

Use the *-from_pin* option to choose between multiple input pins with arcs to this output pin. This option is required.

`-rise`

Sets driving cell information for only the rising port transition.

`-fall`

Sets driving cell information for only the falling port transition.

`-min`

Sets driving cell information for only the minimum analysis. This option is valid only in min-max mode.

`-max`

Sets driving cell information for only the maximum analysis. This option is valid even when not in min-max mode. When not in min-max mode, the option is not required because it is the default.

`-library` *lib_name*

Specifies the name of the library containing the *library_cell_name* value. This is the library of the driving cell. By default, the libraries specified with the *link_path* variable are searched for the cell.

`-pin` *pin_name*

Specifies the output pin whose drive is used. This is the driving pin name. If you do not include the *-from_pin* option, the command uses an arbitrary pin arc ending at the specified pin.

`-from_pin` *from_pin_name*

Specifies an input pin on the specified cell so the command uses the drive of the timing arc from this pin to the specified pin.

`-no_design_rule`

Prevents the transfer of design rules from the driving cell to the port. If you do not include this option, the design rules (such as *max_capacitance*) of the library pin are applied to the port.

`-input_transition_rise` *`rtran`*

Specifies the input rising transition time associated with the *-from_pin* option; if you do not include this option, the default is 0. Use the *-input_transition_rise* and *-input_transition_fall* options to capture the accurate transition time associated with the *from_pin_name* value. This can obtain more accurate information on the transition time and delay time at the output pin.

`-input_transition_fall` *`ftran`*

Specifies the input falling transition time associated with the *from_pin_name* value; if you do not include this option, the default is 0.

`-clock` *`clock_name`*

Specifies the clock related to the driving cell. This option is deprecated and ignored by the tool.

`-clock_fall`

Specifies that driving cell is relative to the falling edge of the clock. The default is the rising edge. This option is deprecated and ignored by the tool.

`-sms_scenarios`

Limits the specification to only the provided SMS scenarios. When this option is not given, the specification applies to all SMS scenarios. Use the *get_sms_scenarios* command to obtain SMS scenarios.

Only input transition times (*-input_transition_rise* and *-input_transition_fall* support this option. For other options, such as *-lib_cell*, the last value applies to all SMS scenarios.

`port_list`

Provides a list of input ports. The list contains input or inout port names in the current design on which the driving cell information is set.

Description

The *set_driving_cell* command sets the port driving cell. It sets attributes on the specified input or inout ports in the current design to associate an external driving cell with the ports. The drive capability of the port is the same as if the specified driving cell were connected in the same context to allow accurate modeling of port drive capability for nonlinear delay models. Use the *report_port* command with the *-drive* option to view drive information on ports.

To automatically set driving cell information on subdesign ports based on their context in the entire design, use the *characterize_context* command. To remove driving cell information from ports, use the *remove_driving_cell* command.

Note: There are two other methods of describing port drive capability, using the *set_drive* or *set_input_transition* command. The most recent drive command has precedence. If possible, always use the *set_driving_cell* command instead of the *set_drive* command because the *set_driving_cell* command allows accurate calculation of port delay and transition time for library cells with nonlinear dependence on capacitance.

Examples

The following example shows how to use the *set_driving_cell* command, and shows how to view drive information on ports using the *report_port* command.

```
pt_shell> set_driving_cell -lib_cell AND [get_ports A]

pt_shell> report_port -drive A
  ...

                 Resistance          Transition
Input Port    Rise      Fall      Rise      Fall
-----------------------------------------------------------
A             --        --        --        --


                     Driving Cell
Input Port    Rise                Fall          Mult    Attrs
-------------------------------------------------------------
A             AND                 AND           --
```

See Also

- all_inputs
- characterize_context
- remove_driving_cell
- report_port
- reset_design
- set_drive
- set_load
- set_input_transition

Feedback

set_eco_options

Specifies options for the *fix_eco_timing*, *fix_eco_drc*, and *fix_eco_power* commands.

Syntax

```
status set_eco_options
   [-physical_tech_lib_path file_name_list]
   [-physical_lib_path file_name_list]
   [-physical_design_path file_name_list]
   [-physical_icc2_lib icc2_lib_directory_path]
   [-physical_icc2_blocks icc2_block_name_list]
   [-physical_constraint_file file_name]
   [-physical_lib_constraint_file file_name_list]
   [-log_file file_name]
   [-mim_group cell_list]
   [-filler_cell_names list]
   [-programmable_spare_cell_names list]
   [-drc_setup_margin margin]
   [-drc_hold_margin margin]
   [-timing_setup_margin margin]
   [-timing_hold_margin margin]
   [-power_setup_margin margin]
   [-power_hold_margin margin]
   [-physical_enable_clock_data]
   [-physical_enable_all_vias]
   [-keep_site_names site_name_list]
   [-convert_sites site_name_list]
   [-power_ground_net_layer pg_layer_name]
   [-allow_missing_lef macro_name_list]
   [-cell_em_profile profile_name]
   [-enable_pin_color_alignment_check]
   [-enable_via0_alignment_check]
```

Data Types

file_name_list	list
icc2_lib_directory_path	string
icc2_block_name_list	list
file_name	string
cell_list	list
list	list
margin	float
site_name_list	list
pg_layer_name	list
macro_name_list	list
profile_name	string

Feedback

Arguments

`-physical_tech_lib_path` *`file_name_list`*

Specifies a list of technology files in Library Exchange Format (LEF). The tool uses the technology LEF files to understand the physical constraints that apply to the cell LEF files, including the layer definitions, via definitions, via rules, and overlapped layer constructs used to specify polygon shapes for LEF macros.

If multiple LEF files are being used, be sure to specify the technology LEF files before the cell LEF files in the file name list. Otherwise, physically aware ECO commands will not have the correct physical constraint definitions, leading to incorrect physical ECO results.

`-physical_lib_path` *`file_name_list`*

Specifies a list of physical library files in Library Exchange Format (LEF). The tool uses the physical library data to understand the physical constraints such as the physical shapes of pins, cells, and blocks.

For more information, see "Specifying LEF/DEF Files" in the EXAMPLES section.

`-physical_design_path` *`file_name_list`*

Specifies a list of physical data files in Design Exchange Format (DEF). The tool uses the physical design data to understand the design layout details such as available free sites and area utilization density.

The specified DEF files must exist, and the *DESIGN_NAME* fields in them must match block names in the current design. If a block already has DEF associated with it, this new specification overwrites the previous specification.

If this option is used together with the *-physical_constraint_file* option, only a single DEF file can be specified.

`-physical_icc2_lib` *`icc2_lib_directory_path`*

Specifies an IC Compiler II reference library directory path. The tool obtains all technology and cell LEF information from reference libraries in this directory. If you need additional LEF information not available in these reference libraries, you can also use the *-physical_lib_path* option.

Each *set_eco_options* command accepts no more than one physical IC Compiler II reference library directory path. However, you can use multiple *set_eco_options* commands to specify multiple reference library directory paths for the session.

`-physical_icc2_blocks` *`icc2_block_name_list`*

Specifies a list of IC Compiler II block names. The PrimeTime tool reads in the physical block data from the library directory path specified by the *-physical_icc2_lib* option. The tool reads in only the physical data for the listed blocks.

`-physical_constraint_file` *`file_name`*

Specifies a file that contains *create_voltage_area* or *create_placement_blockage* commands. These commands specify the following physical constraints:

- Voltage areas - In a design with multiple voltage areas, the tool needs to know their locations so it can place each buffer in the correct area. To specify voltage areas, include *create_voltage_area* commands in the physical constraint file.
- Placement blockages - To prevent buffer insertion and cell resizing in specific areas, specify the placement blockage areas by including *create_placement_blockage* commands in the physical constraint file.

Failing to include necessary *create_voltage_area* and *create_placement_blockage* commands in the physical constraint file might result in poor QoR or incorrect results.

The physical constraint file is applied to the physical block specified by the *-physical_design_path* DEF file. As a result, only a single DEF file can be specified at a time when a physical constraint file is also specified.

Physical constraint file specifications remain on a block until replaced or explicitly removed with an empty string (*-physical_constraint_file {}*) specification.

`-physical_lib_constraint_file` *`file_name_list`*

Specifies a list of physical library constraint files. A file can contain user-specified *set_lib_cell_spacing_label* and *set_spacing_label_rule* commands to specify intercell spacing labels and rules. It can also contain advanced-node spacing constraints for objects on the Vth implant, diffusion, and poly layers. It can be a user-written ASCII text file or a binary file written by the *export_advanced_technology_rules* command of the IC Compiler or IC Compiler II tool.

`-log_file` *`file_name`*

Specifies the name of a log file. The tool writes warning and error messages to this file when it loads the physical library and physical design files. Review the log file if the tool encounters any issues during physical ECO fixing.

`-mim_group` *`cell_list`*

Specifies a list of cells to be treated as a set of multiply instantiated module (MIM) instances that share a similar context. This option has an effect only if the *eco_enable_mim* variable is set to *true*.

When the *eco_enable_mim* variable is set to *true*, the tool automatically detects multiple instances of the same cell and puts those cells into a MIM group. The *write_changes* command assumes that the members of a MIM group have similar context, and therefore creates a single change list that applies to all members of the MIM group. You can use the *-mim_group* option to separate a large MIM group into smaller MIM groups, each having a different context and a separate change list.

No actual changes are made to the design until you run an ECO command: *fix_eco_timing*, *fix_eco_drc*, or *fix_eco_power*. As a result, the tool reports any MIM specification errors at that time.

You can view the current MIM configuration by using the *report_eco_options* command. You can use the *reset_eco_options* command to remove it if no ECO changes have been made to the design.

For more information, see "Specifying Multiply Instantiated Module (MIM) Groups" in the DESCRIPTION section.

`-filler_cell_names` *`list`*

Specifies the names of LEF library cells that are to be treated as filler cells by physically aware ECO commands, even when the cells do not meet the filler cell LEF requirements. For details, see "Filler Cells" in the EXAMPLES section.

`-programmable_spare_cell_names` *`list`*

Specifies a list of programmable spare cell leaf names for the freeze silicon physical ECO flow. Each programmable spare cell in the list must be:

- Defined in the .lib library files
- Defined as a MACRO in the LEF files specified by the *-physical_lib_path* option
- Instantiated as + *SOURCE DIST* + *PLACED* components in the DEF files specified by the *-physical_design_path* option

  Do not use wildcard characters to specify the leaf names.

  For more information, see "Programmable Spare Cells" in the EXAMPLES section.

`-drc_setup_margin` *`margin`*

Specifies the amount of setup slack to maintain during DRC fixing by the *fix_eco_drc* command. The default is negative infinity; that is, the *fix_eco_drc* command fixes violations no matter how much setup slack is degraded. Specify the margin in library time units.

`-drc_hold_margin` *`margin`*

Specifies the amount of hold slack to maintain during DRC fixing by the *fix_eco_drc* command. The default is negative infinity; that is, the *fix_eco_drc* command fixes violations no matter how much hold slack is degraded.

`-timing_setup_margin` *`margin`*

Specifies the amount of setup slack to maintain during timing fixing by the *fix_eco_timing* command. The default is zero. Specify the margin in library time units.

`-timing_hold_margin` *`margin`*

Specifies the amount of hold slack to maintain during timing fixing by the *fix_eco_timing* command. The default is zero.

`-power_setup_margin` *`margin`*

Specifies the amount of setup slack to maintain during power recovery by the *fix_eco_power* command. The default is zero. Specify the margin in library time units.

`-power_hold_margin` *`margin`*

Specifies the amount of hold slack to maintain during buffer removal by the *fix_eco_power* command. The default is zero. This option applies only to buffer removal, not cell swapping or downsizing.

`-physical_enable_clock_data`

Enables usage of clock network physical data for ECO purposes. Without this option, ECO operations ignore physical data for the clock network.

`-physical_enable_all_vias`

Enables reading of physical data for all vias. Without this option, the tool ignores physical data for all vias.

`-keep_site_names` *`site_name_list`*

Specifies a list of DEF cell row site names. When reading physical data, the tool saves the corresponding rows defined in DEF. By default, the tool saves all cell rows that have legal site definitions in LEF.

`-convert_sites` *site_name_list*

Specifies the conversion of the DEF and LEF site names in physically aware ECO flows. Use the following syntax to specify the site name conversion:

`-convert_sites {{`*old_site1 new_site_1*`} {`*old_site2 new_site2*`} ...}`

Use this option when the LEF site names do not match the DEF site names or identical sites are defined with different site names in multiple DEF or LEF files. When the tool reads in the physical data for the design, any site names that match the old site names of site_name_list are converted to the new site names in DEF and LEF files.

`-power_ground_net_layer` *pg_layer_name*

Specifies the power/ground (PG) net layer name for power/ground (PG) straps. These straps are typically vertical on bottom metal layers.

With this option, the tool keeps information for PG straps on the specified layer when the physical data is read. ECO commands (*fix_eco_timing*, *fix_eco_drc*, and *fix_eco_power*) use this saved data during physically aware ECO to avoid DRC violations between PG straps and signal pins.

Without this option, PG straps will not be known to ECO fixing. This can lead to physical DRC violations caused by ECO cell signal pin proximity to PG straps in the design, resulting in cell displacements or routing challenges.

Note: This option requires a PrimeTime-ADV-PLUS license.

`-allow_missing_lef`

Specifies a list of model names that are allowed to have missing LEF macros. Components associated with these model names are ignored by physical ECO commands, instead of causing them to fail.

`-cell_em_profile`

Specifies the name of cell electromigration (EM) lifetime profile to be considered during cell EM fixing (*fix_eco_drc -type cell_em*). Only those EM violations will be considered whose profile name matches with that provided with this option. By default, all EM violations will be considered for fixing. To reset a previously specified value back to default behavior, behavior, specify a value of *default* with this option.

`-enable_pin_color_alignment_check`

Enables checking for proper color alignment of cell internal pins and metal shapes with routing tracks during ECO optimization. Without this option, ECO optimization does not perform the pin color alignment check.

Note: This option requires a PrimeTime-ADV-PLUS license.

`-enable_via0_alignment_check`

Enables checking for proper alignment of cell PG pin internal via0 shapes with the via0 grid during ECO optimization. Without this option, ECO optimization does not perform the via0 alignment check.

Description

The *set_eco_options* command specifies the files that contain physical data for physical ECO commands: *fix_eco_timing*, *fix_eco_drc*, and *fix_eco_power*. You must set the ECO options before you run the physical ECO commands.

Like the *link_path* variable that specifies the files to be linked, the files specified by *set_eco_options* command are not loaded into memory when you run the command. Instead, a physical ECO command (such as *fix_eco_timing*) identifies the files specified by the *set_eco_options* command and loads them into memory before fixing begins.

Distributed Multi-Scenario Analysis

To run the *set_eco_options* command in the DMSA flow, use the *remote_execute* command to run it in the worker processes. You cannot run the *set_eco_options* command in the manager process.

Specifying Multiply Instantiated Module (MIM) Groups

When the *eco_enable_mim* variable is set to *true*, the tool automatically detects multiple instances of the same cell and puts those cells into a MIM group. The *write_changes* command assumes that the members of a MIM group have similar context, and therefore creates a single change list that applies to all members of the MIM group.

You can break an existing MIM group into multiple groups or specify a MIM group for a flat parasitics file by using the *-mim_group* option of the *set_eco_options* command. Note that if there is an existing MIM group, you cannot specify a new MIM group or add new instances to the MIM group; you can only break a MIM group into smaller MIM groups.

Suppose a module called CPU is instantiated five times in the TOP design as CPU1 through CPU5, and there are CPU.def and CPU.SPEF files associated with the CPU module. Also you have the following command in your script:

```
read_parasitics -path {CPU1 CPU2 CPU3 CPU4 CPU5} CPU.SPEF
```

If the *eco_enable_mim* variable is set to *true*, when you use the *fix_eco_timing*, *fix_eco_drc*, or *fix_eco_power* command, the tool detects the CPU module as a MIM. However, you might want to break the 5-instance MIM three groups, {CPU1 CPU2}, {CPU3 CPU4}, and {CPU5}, because of large timing differences between the three groups. In this case, use the following script:

```
set_eco_options -mim_group {CPU1 CPU2}
set_eco_options -mim_group {CPU3 CPU4}
```

The instance CPU5 implicitly belongs to its own MIM group, so it is effectively treated as an independent, non-MIM instance.

Suppose the CPU instances are now grouped into three MIM groups: {CPU1 CPU2}, {CPU3 CPU4} and {CPU5}. Suppose each CPU module also includes two instances of the ALU module, ALU1 and ALU2. The ALU instances are not treated as MIMs because regrouping instances in different MIM groups is not allowed, and creating new MIM groups is not allowed.

Suppose that you have a hierarchical netlist but a single flat parasitic data file. In this case, the tool cannot infer MIM groups from identical parasitic data, even when the *eco_enable_mim* variable is set to *true*. However, you can explicitly specify the MIM groups as follows:

```
set_eco_options -mim_group {CPU1 CPU2 CPU3 CPU4 CPU5}
```

Suppose the CPU module includes an ALU module, which is also a MIM. This is an example of a nested MIM hierarchy. The following commands specify this configuration to recognize both the CPU and ALU modules as MIMs.

```
set_eco_options -mim_group [all_instances -hierarchy CPU]
set_eco_options -mim_group [all_instances -hierarchy ALU]
```

The *set_eco_options* commands must define the MIM groups in order, from higher to lower in the hierarchy. Reversing the order of the two commands would not work correctly.

Examples

Specifying LEF/DEF files

Specify LEF files with the *-physical_lib_path* option and DEF files with the *-physical_design_path* option:

```
pt_shell> set lef_files      [glob $my_physical_data/lef/cell_lef_*.lef]
pt_shell> set tech_lef_files [$my_physical_data/lef/technology.lef]
pt_shell> set_eco_options -physical_tech_lib_path $tech_lef_files \\
                          -physical_lib_path $lef_files \\
                          -physical_design_path design.def.gz \\
                          -log_file ./error.log
```

The following script specifies the same information in a distributed multi-scenario analysis environment.

```
remote_execute -verbose {
  set lef_files      [glob $my_physical_data/lef/cell_lef_*.lef]
  set tech_lef_files [$my_physical_data/lef/technology.lef]
  set_eco_options -physical_tech_lib_path $tech_lef_files
                  -physical_lib_path $lef_files
                  -physical_design_path design.def.gz
                  -log_file ./error.log
}
```

Overwriting DEF and physical constraint file specifications

Specify a DEF file and its associated physical constraint file:

```
pt_shell> set_eco_options -physical_design_path     design.def.gz \\
                          -physical_constraint_file phy_constr.tcl
```

Replace the previous DEF specification by a new DEF file for the same design, while preserving the physical constraint file specification:

```
pt_shell> set_eco_options -physical_design_path
  same_design.new_def.gz
Warning: Overwriting previously defined physical data specification for
 block 'design'.
```

Replace both specifications by new files:

```
pt_shell> set_eco_options -physical_design_path
  same_design.new_def.gz \\
                          -physical_constraint_file new_phy_constr.tcl
Warning: Overwriting previously defined physical data specification for
 block 'design'.
```

Specifying Physical Data From the IC Compiler II Database

To specify technology and cell LEF information from the IC Compiler II tool, use the *-physical_icc2_lib* option; and to specify the physical block design information, use the *-physical_icc2_blocks* option:

```
pt_shell> set icc2_lib_directory_path $my_icc2_ndm_lib_directory
pt_shell> set icc2_blocks $my_block
pt_shell> set_eco_options -physical_icc2_lib $icc2_lib_directory_path \\
                             -physical_icc2_blocks $my_block \\
                             -log_file ./error.log
```

Labels may be used to distinguish between alternate versions of a block. To do so, specify the label name after the block name, separated by a slash:

```
pt_shell> set_eco_options -physical_icc2_lib $icc2_lib_directory_path \\
                             -physical_icc2_blocks $my_block/$my_label \\
                             -log_file ./error.log
```

Filler Cells

PrimeTime physically aware ECO commands follow LEF and DEF rules to identify filler cells. Both the LEF and DEF requirements must be met for components to be treated as filler cells.

To identify a filler cell macro, the LEF macro must have its CLASS CORE type specified as SPACER or FEEDTHRU, as shown in the following example.

```
MACRO FILL1TLL
 CLASS CORE SPACER ;
 ORIGIN 0 0 ;
 SIZE 6.4 BY 1.6 ;
```

To identify a filler cell component, the DEF component must be a physical-only cell as specified by the + SOURCE DIST construct in the component definition. In addition, the filler cell component must be marked as + PLACED, must have valid physical coordinates and orientation specified, and must not be present in the Verilog netlist. For example,

```
xofiller!FILL4TLL!11037 FILL43TLL + SOURCE DIST + PLACED ( 264300
 624100 ) N ;
```

Filler cell components marked as + FIXED or + COVER cannot be used for ECO and are treated as permanent occupied sites.

You can relax the LEF requirements for filler cells by using the *-filler_cell_names* option of the *set_eco_options* command. With this option, you can specify any LEF macro to be a filler cell. For example,

```
set user_filler_cells "FILL4TLL"
set_eco_options -physical_lib_path $lef_files
                -physical_design_path design.def.gz
                -filler_cell_names $user_filler_cells
                -log_file ./error.log
```

All automatically detected and user-specified LEF filler cell macros are reported in the LEF/DEF log file as shown in the following examples.

```
Information: Identified auto filler cell FILL8TLL at lineNumber 10.
Information: Identified user filler cell FILL4TLL at lineNumber 11.
```

The DEF filler cell requirements must be met, even if you relax the LEF filler cell requirements. The physically aware ECO commands treat all components that meet the LEF and DEF requirements as filler cells. If the *eco_allow_filler_cells_as_open_sites* variable is set to *true*, physically aware ECO commands treat all filler cell components as open sites.

Programmable Spare Cells

The following example specifies programmable spare cells in the LEF/DEF files.

To identify a programmable spare cell macro, the LEF macro must be defined in one of the LEF files specified by the *-physical_lib_path* option; there are no other LEF requirements. For example,

```
MACRO FILL4TLL
 CLASS CORE ;
 ORIGIN 0 0 ;
 SIZE 6.4 BY 1.6 ;
```

To identify a programmable spare cell component, the DEF component must be a physical-only cell as specified by the + SOURCE DIST construct in the component definition. In addition, the programmable spare cell component must be marked as + PLACED and must have valid physical coordinates and orientation specified. For example,

```
xofiller!FILL4TLL!lvt!11037 FILL4TLL + SOURCE DIST + PLACED ( 264300
 624100 ) N ;
```

Spare cell components marked as + FIXED or + COVER are not treated as programmable spare cells in the freeze silicon physical ECO flow.

To report all user-specified programmable spare cells names, use the *report_eco_options* command. All LEF programmable spare cell macros detected by the tool are reported in the LEF/DEF log as shown in the following example:

```
Information: Identified programmable spare cell FILL4TLL at lineNumber
 10.
```

Specifying Physical Library Spacing Rule Constraints

The following example specifies a physical library constraint file containing spacing rules.

```
set rule_file spacing_rules.tcl
set_eco_options -physical_lib_path $lef_files
                -physical_design_path design.def.gz
                -physical_lib_constraint_file $rule_file
                -log_file ./error.log
```

Enabling Physically Aware Clock Network ECOs

The following example enables reading physical data for the clock network.

```
pt_shell> set_eco_options -physical_enable_clock_data
```

Enabling Physical Loading of All Vias

The following example enables reading physical data for all vias.

```
pt_shell> set_eco_options -physical_enable_all_vias
```

Specifying Site Names

The following example enables saving DEF rows defined with the site name "unit1" when reading physical data.

```
pt_shell> set_eco_options -keep_site_names {unit1}
```

Converting DEF Sites

The following example converts DEF rows of site "unit" into rows of site "CORE2T".

```
pt_shell> set_eco_options -convert_sites {{unit CORE2T}}
```

This specifies the mapping of site names from the DEF to the LEF file:

```
In DEF file:

ROW STD_ROW_324 unit 560 85680 FS ...;
ROW STD_ROW_325 unit 560 88200 N ...;

In LEF file:

SITE CORE2T
  CLASS CORE ;
  SYMMETRY X Y ;
  SIZE 0.14 BY 1.2 ;
END CORE2T
```

Specifying Power Ground Net Layer

The following example enables PG aware ECO and saves physical data for PG on layer M1.

```
pt_shell> set_eco_options -power_ground_net_layer M1
```

Allowing Missing LEF Macros

The following example allows components to be associated with two model names that are not present as LEF macros.

```
pt_shell> set_eco_options -allow_missing_lef { MODEL1 MODEL2 }
```

Specifying Cell EM Lifetime Profile name

The following example sets profile_a as profile to be considered during cell EM fixing

```
pt_shell> set_eco_options -cell_em_profile profile_a
```

Enabling Cell Pin Color Alignment Checking

The following example enables color alignment checking during ECO optimization for cell-internal pins and metal shapes.

```
pt_shell> set_eco_options -enable_pin_color_alignment_check
```

Enabling via0 Alignment Checking

The following example enables via0 alignment checking during ECO optimization.

```
pt_shell> set_eco_options -enable_via0_alignment_check
```

See Also

- create_placement_blockage
- create_voltage_area

Feedback

- fix_eco_drc
- fix_eco_timing
- fix_eco_power
- report_eco_options
- reset_eco_options
- set_lib_cell_spacing_label
- set_spacing_label_rule
- write_changes
- eco_enable_mim
- eco_physical_match_site_row_names
- eco_enable_overwrite_physical_design_path

---

set_eco_scenarios

Controls how violation types are considered across scenarios during live and static scenario analysis.

Syntax

```
status set_eco_scenarios
[-only]
[-ignore]
-type fix_type_list
scenario_name_list
```

Data Types

```
scenario_name_list   list
fix_type_list        list
```

Arguments

`-only`

Specifies that in the *scenario_name_list* scenarios, only the *fix_type_list* violation types will be considered by live/static scenario selection. Other violation types will be ignored.

Feedback

`-ignore`

Specifies that in the *scenario_name_list* scenarios, the specified *fix_type_list* violation types will be ignored by live/static scenario selection. Other violation types will be considered.

`-type` *`fix_type_list`*

Specifies a list of one or more timing violation types: *setup*, *hold*, *max_transition*, *max_capacitance*, or *noise*.

`scenario_name_list`

Specifies one or more scenarios to which the specification applies.

Description

The *set_eco_scenarios* command allows you to control what data is considered for live and static scenario analysis and selection in the hybrid ECO flow. For example, you could ignore setup slacks in certain hold-only scenarios.

The *-only* and *-ignore* options are mutually exclusive; one of them must be specified.

Note that this command affects only the live and static scenario analysis performed during the *start_eco_scenarios* or *report_eco_scenarios* command; it does not affect what data is used by the *fix_eco_** commands themselves.

Examples

The following example considers only setup timing violations in ss1, ss2, ss3 and ss4 scenarios:

```
pt_shell> set_eco_scenarios -only -type {setup} {ss1 ss2 ss3 ss4}
```

The following example ignores setup timing violations in ff1, ff2, ff3 and ff4 scenarios:

```
pt_shell> set_eco_scenarios -ignore -type {setup} {ff1 ff2 ff3 ff4}
```

See Also

- reset_eco_scenarios
- report_eco_scenarios
- start_eco_scenarios

---

set_em_analysis_options

Sets the options for cell EM analysis.

Feedback

Syntax

status *set_em_analysis_options*

```
[-in_slew slew_type]
[-cload_type load_type]
[-cload_val load_val]
[-current_type current_type]
```

Data Types

```
slew_type          max | min | avg
load_type          tot | eff
load_val           max | min | avg
current_type       rms | peak | ave
```

Arguments

`-slew_val max | min | avg`

Specifies slew value to be used in cell EM analysis. Default slew value is avg.

`-cload_type tot | eff`

Specifies load type to be used in cell EM analysis. Default cload type is effective.

`-cload_val max | min | avg`

Specifies load value to be used in cell EM analysis. Default load value is avg.

`-current_type rms | peak | ave`

Specifies that EM analysis options will be applied to spesific current type.

Description

Command to specify various analysis options for cell EM analysis in PrimePower. The options from this command will control how cell EM analysis behaves for specific analysis type.

Examples

The following example will use effective max capacitance, min slew for rms type cell EM analysis.

```
pt_shell> > set_em_analysis_options -in_slew min -cload_val max
 -cload_type eff -current_type rms
```

See Also

- get_em_max_toggle_rate
- report_cell_em_violation

Feedback

set_em_scaling_factor

Sets EM scaling factors for specified current type on a library cell EM arc.

Syntax

status *set_em_scaling_factor*

```
[-input_pin input_pin]
[-output_pin output_pin]
[-factor factor]
[-current_type current_type]
object
```

Data Types

```
input_pin         list
output_pin        list
factor            float
current_type      rms | peak | ave
object               list
```

Arguments

`-input_pin list`

Specifies input library pin name for EM arc.

`-output_pin list`

Specifies output library pin name for EM arc.

`-factor float`

Specifies cell EM scaling factor that is to be applied on the specified arc.

`-current_type rms | peak | ave`

Specifies scaling factor to be set for specific current type on an arc. The current type are ave, rms, peak. Default it will applied to all current type.

`object list`

Specifies library cells for which the EM scaling factor is applied.

Description

The command allows user to set a scaling factor for max toggle rate looked up from library cell EM arc. Scaling factor can be EM arc specific and current type specific. The scaling factor is applied on original max toggle rate from library.

Feedback

Examples

```
pt_shell> > set_em_scaling_factor -input_pin  a -output_pin  a
 -current_type ave -factor 0.5 tech_3.3v_25t_1p/buf1
```

See Also

- get_em_max_toggle_rate
- report_cell_em_violation

---

set_emmp_configuration

Specifies the configuration options to run estimate_memory_max_power command for a hierarchical cell and groups of leaf cell.

Syntax

status *set_emmp_configuration*

```
hier_instance_name
-list leaf_instance_group_list
```

Data Types

```
hier_instance_name           string
leaf_instance_group_list     list
```

Arguments

hier_instance_name

To specify the hierarchical cell for which the maximum power estimation is to be done.

leaf_instance_group_list

To specify the list of groups, the groups specify list of leaf cell.

Description

The *set_emmp_configuration* command specifies the logical memory configuration for a hierarchical memory cell which has leaf level memory cells. The user can specify groups of leaf memory cells. The activity on hierarchy memory cell will be shared between groups. The user must set this configuration before estimate_memory_max_power analysis with state_derate_pairs options.

Examples

The following examples shows set memory configuration. Here mem is the hierarchical cell and p1, p2, p3 and p4 are leaf level memory cells inside "mem".

Feedback

```
pt_shell> set_emmp_configuration mem -list "{p1 p2} {p3 p4}"
```

See Also

- reset_emmp_configuration
- estimate_memory_max_power

---

set_equal

Sets two ports to be logically equivalent.

Syntax

```
status set_equal
  port1
  port2
```

Data Types

```
port1        string
port2        string
```

Arguments

port1

Specifies the first input port.

port2

Specifies the second input port.

Description

Defines two input ports in the current design as logically equivalent. This information is used during synthesis to eliminate redundant ports, improving optimization quality.

Logical inconsistencies are checked for in relationships between ports. Specifying an inconsistent logical relationship between ports is not allowed.

To remove this property from a port, run the *reset_design* command.

This attribute is characterized by the *characterize_context* command and exported to synthesis using the *write_context* command.

Examples

The following example sets two input ports "A" and "B" logically equal.

```
pt_shell> set_equal A [get_ports B]
1
```

The following example sets up a contradictory relationship between two ports and creates an error.

```
pt_shell> set_equal A B
1
pt_shell> set_opposite A B
Error: Can't set equal ports opposite in design 'top': 'A' 'B'. (DDB-22)
0
```

See Also

- characterize_context
- set_opposite
- write_context

---

set_equivalent

Declares a set of supply nets to be treated as equivalent.

Syntax

string *set_equivalent*

```
-nets list_of_supply_nets
-reset
```

Data Types

list_of_supply_nets	list
reset	boolean

Arguments

`-nets list_of_supply_nets`

Specifies the list of supply nets declared equivalent by this command.

`-reset`

Reset the equivalence of the supply nets specified with optional -nets argument. If -nets is not specified, all equivalences set before are removed. All other arguments are invalid if this option is specified.

`-primary_net supply_net name`

Specifies the supply net that is to refer all other nets declared equivalent by this command.

Feedback

Description

The *set_equivalent* command declares a list of supply nets to be equivalent. As a result, the supply nets are considered to be always at the same voltage. The *set_voltage* and *set_voltage_levels* commands apply to all the supply net segments of the equivalent supply nets. For simultaneous multivoltage analysis (SMVA), no domain crossing is inferred between the equivalent supply nets.

The behavior of *set_equivalent* is analogous to the connections inferred through UPF power switches when the *upf_power_switches_always_on* variable is set to *true*.

Examples

The following command declares the supply nets VDD, VDDS, VDD2 and VDD2S to be equivalent:

```
pt_shell> set_equivalent -nets {VDD VDDS VDD2 VDD2S}
1
```

The following command declares the supply nets VDDI, VDDIS and VDDW to be equivalent, and it also sets VDDW to be the supply group that refers to them:

```
pt_shell> set_equivalent -nets {VDDI VDDIS VDDW} -primary_net VDDW
1
```

Note that the *set_equivalent* command does not create a supply group for each equivalent net. The *-primary_net* option must be used to identify a single net to be used as the supply group reference net, which can then be referenced by other commands such as *set_voltage_levels*.

See Also

- set_voltage
- set_voltage_levels
- upf_power_switches_always_on

---

set_essential_map

Sets the dumping of essential mapping files.

Syntax

int *set_essential_map*

option

Feedback

Data Types

option bool

Arguments

option

Specifies whether to write out the essential or non-essential mapping files.

Description

The *set_essential_map* writes out the essential and non-essential mapping files in the PrimePower RTL tool. Essential mapping file: Contains mapping for all essential design nodes such as primary inputs, sequential outputs, memory, instantiated clock gate cell outputs and black-box pins. Non-essential mapping file: Contains map info only for design nodes that underwent a name change. The default is true, the RTL Architect tool writes out an essential mapping file. When set to false, the RTL Architect writes out the non-essential map file.

Examples

This example writes out the essential mapping file

```
pwr_shell> set_essential_map true
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_phy_tech_file
- set_synth_output_dir
- set_rtl_architect_shell
- set_presynth_options
- set_synth_options
- set_postsynth_options

---

set_extract_model_indexes

Sets indexes on specified ports for extracted timing model (ETM) generation.

Syntax

```
status set_extract_model_indexes
  [-ports port_list]
```

```
  -type transition | capacitance
  [-min]
  [-max]
  [-rise]
  [-fall]
  index_list
```

Data Types

```
port_list      list
index_list     list
```

Arguments

`-ports port_list`

Specifies a list of ports on which to set the index values. If you do not specify this option, the command sets index values on all ports by default.

`-type transition | capacitance`

Applies the *index_list* to transition (slew) or capacitance (load) tables in the ETM library.

`-min`

Applies the *index_list* to minimum arcs only. If you specify neither *-min* nor *-max*, the command applies the index values to both minimum and maximum arcs.

`-max`

Applies the *index_list* to maximum arcs only. If you specify neither *-min* nor *-max*, the command applies the index values to both minimum and maximum arcs.

`-rise`

Applies the *index_list* to rise timing tables only. If you specify neither *-rise* nor *-fall*, the command applies the index values to both rise and fall tables.

`-fall`

Applies the *index_list* to fall timing tables only. If you specify neither *-rise* nor *-fall*, the command applies the index values to both rise and fall tables.

index_list

Specifies a list of index values (floating) to be applied on specified ports. The index values should be positive and monotonically increasing.

Description

This command replaces automatically generated indexes with the user-specified indexes on specific ports.

If the *extract_model* command detects index values that violate maximum and minimum bounds set in the library for max_transition and max_capacitance, automatic indexing is used for those ports.

See Also

- extract_model
- reset_extract_model_indexes

---

set_extract_model_margin

Sets margins to be applied to specified ports in extracted timing model (ETMs).

Syntax

```
status set_extract_model_margin
 [-ports port_list]
 [-min]
 [-max]
 margin_value
```

Data Types

```
port_list    list
margin_value   float
```

Arguments

`-ports port_list`

Specifies a list of ports on which to apply the specified margin. If you do not specify this option, the command sets the margin on all ports by default.

`-min`

Applies the *margin_value* to minimum arcs only. If you specify neither *-min* nor *-max*, the command applies the margin to both minimum and maximum arcs.

`-max`

Applies the *margin_value* to maximum arcs only. If you specify neither *-min* nor *-max*, the command applies the margin to both minimum and maximum arcs.

margin_value

Specifies the absolute margin as a positive floating point number in units of main library. The margin is applied after ETM generation.

Feedback

Description

This command applies a margin value to offset the values in the table. For maximum arcs, the margin is added to already computed delay values. For minimum arcs, the margin is subtracted from the delay values.

To reset the margin values, specify a margin value of 0.

See Also

- extract_model

---

set_false_path

Identifies paths in a design that are to be marked as false, so that they are not considered during timing analysis.

Syntax

status *set_false_path*

```
[-setup]
[-hold]
[-rise]
[-fall]
[-reset_path]
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-comment comment_string]
[-sms_scenarios sms_scenarios_list]
```

Data Types

comment_string	string
fall_from_list	list
fall_through_list	list
fall_to_list	list
from_list	list
rise_from_list	list
rise_through_list	list
rise_to_list	list
sms_scenarios_list	collection
through_list	list
to_list	list

Feedback

Arguments

`-setup`

Indicates that setup (maximum) paths are to be marked as false. This option disables setup checking for specified paths. If you do not specify either the *-setup* or *-hold* option, both setup and hold timing are marked false.

`-hold`

Indicates that hold (minimum) paths are to be marked as false. This option disables hold checking for specified paths. If you do not specify either the *-setup* or *-hold* option, both setup and hold timing are marked false.

`-rise`

Indicates that rising delays are to be marked as false, as measured on the path endpoint. If you do not specify either the *-rise* or *-fall* option, both rise and fall timing are marked false.

`-fall`

Indicates that falling delays are to be marked as false, as measured on the path endpoint. If you do not specify either the *-rise* or *-fall* option, both rise and fall timing are marked false.

`-reset_path`

Indicates that existing point-to-point exception information is to be removed from the specified paths. If used with only the *-to* option, all paths leading to the specified endpoints are reset. If used with only the *-from* option, all paths leading from the specified startpoints are reset. If used with the *-from* and *-to* options, only paths between those points are reset. Only information of the same rise/fall setup/hold type is reset. This is equivalent to using the *reset_path* command with similar arguments before the *set_false_path* command is issued.

`-from` *`from_list`*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If you specify a cell, one path startpoint on that cell is affected.

You can use only one of the *-from*, *-rise_from*, or *-fall_from* options.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the

clock source, taking into account any logical inversions along the clock path. You can use only one of the *-from*, *-rise_from*, or *-fall_from* options.

`-fall_from` *`fall_from_list`*

Same as the *-from* option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the *-from*, *-rise_from*, or *-fall_from* options.

`-through` *`through_list`*

Specifies a list of pins, ports, cells or nets through which the disabled paths must pass. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you do not specify any type of *through* option, all timing paths specified using the *-from* and *-to* options are affected.

You can use multiple *-through*, *-rise_through*, and *-fall_through* options in a single command to specify paths that traverse through multiple points in the design. The tool respects the order in which you specify these options.

The following example specifies paths beginning at A1, passing through B1, then through C1, and ending at D1.

```
-from A1 -through B1 -through C1 -to D1
```

If you specify more than one object with one *-through*, *-rise_through*, or *-fall_through* option, the path can pass through any of the objects. The following example specifies paths beginning at A1, passing through either B1 or B2, then passing through either C1 or C2, and ending at D1.

```
-from A1 -through {B1 B2} -through {C1 C2} -to D1
```

`-rise_through` *`rise_through_list`*

The same as the *-through* option, but the paths must have a rising transition at the through points.

`-fall_through` *`fall_through_list`*

The same as the *-through* option, but the paths must have a falling transition at the through points.

`-to` *`to_list`*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and

primary outputs related to that clock are used as path endpoints. If you specify a cell, one path endpoint on that cell is affected.

You can use only one of the *-to*, *-rise_to*, or *-fall_to* options.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options.

`-fall_to` *`fall_to_list`*

Same as the *-to* option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the *-to*, *-rise_to*, or *-fall_to* options.

`-comment` *`comment_string`*

Associate a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this false path is applied. When this option is not given the false applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

The *set_false_path* command marks startpoint/endpoint pairs as false timing paths; that is, paths that cannot propagate a signal. The command removes timing constraints on these false paths, so that they are not considered during timing analysis. Path startpoints are input ports or register clock pins; path endpoints are register data pins or output ports. The command disables maximum delay (setup) checking and minimum delay (hold) checking for the specified paths.

The *set_false_path* command is a point-to-point timing exception command, and overrides the default single-cycle timing relationship for one or more timing paths. False path information always takes precedence over multicycle path information. Also, the *set_false_path* command overrides the *set_max_delay* and *set_min_delay* commands.

In general, for timing exceptions with different path specifications, the more specific command has priority over the more general one. False paths represent an exception to

this rule. If a timing path is satisfied by two or more false path specifications, the more general one wins. This can be seen with the *report_exceptions -ignored* command.

To disable the timing at a particular cell along a path, use the *set_disable_timing* command.

To remove the false path designations set by the *set_false_path* command, use the *reset_path* command.

For crosstalk analysis the false paths are treated as sensitizable. Therefore, the aggressors on the false paths could still effect its victims. However, the presence of false path could change the borrow behavior in level-sensitive latch designs, potentially borrow latches may no longer borrow leading to change in delta delays for logic nets that are in the fanout of affected latches. When the clocks are asynchronous to each other, the *set_clock_groups -asynchronous* command should be used instead of the *set_false_path* command between the asynchronous clocks. Without the *set_clock_groups -asynchronous* command, the clocks are treated to be synchronous to each other.

When a path has transparent latches in the interior of the path, and there are more than one path specifier (*-from*, *-through*, or \B-to*), set_false_path is only applied when all the path specifiers can be satisfied within one combinational segment between sequential elements. Commands using -from pin, and -to pin, with a transparent latch in between, will be ignored, because both specifiers could not be satisfied in a single combinational segment between sequential elements.*

Examples

The following example disables timing paths from ff12 to ff34.

```
pt_shell> set_false_path -from ff12 -to ff34
```

The following example disables rising timing paths from U14/Z to ff29/Reset.

```
pt_shell> set_false_path -rise_from U14/Z -to ff29/Reset
```

The following examples disables hold checking for endpoints clocked by CLK1.

```
pt_shell> set_false_path -hold -to [get_clocks CLK1]
```

The following example disables all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C}.

```
pt_shell> set_false_path -from ff1/CP -through {U1/Z U2/Z}
            -through {U3/Z U4/C} -to ff2/D
```

The following example disables all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall to one or more of {U3/Z U4/C}.

Feedback

```
pt_shell> set_false_path -from ff1/CP
            -rise_through {U1/Z U2/Z}
            -fall_through {U3/Z U4/C} -to ff2/D
```

See Also

- current_design
- report_exceptions
- reset_design
- reset_path
- set_clock_groups
- set_disable_timing
- set_max_delay
- set_min_delay
- set_multicycle_path

---

set_fanout_load

Sets fanout_load for output ports in the current design.

Syntax

string *set_fanout_load*

```
[-sms_scenarios sms_scenarios_list]
value
port_list
```

Data Types

```
port_list           list
sms_scenarios_list  collection
value               float
```

Arguments

value

Shows the *fanout_load* value, in units consistent with the *fanout_load* and *max_fanout* values in the technology library.

port_list

Provides a list of output ports.

Feedback

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this fanout load is applied. When this option is not given the fanout load applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

Specifies a *fanout_load* for output ports in the current design. By default, ports are considered to have fanout_load of 0.0. Fanout load for a net is the sum of *fanout_load* attributes for all input pins and output ports connected to the net. Output pins may have maximum fanout limits, specified in the library or with the *set_max_fanout* command.

The *report_constraint -max_fanout* command shows maximum fanout constraint evaluations. The *report_port -design_rule* command shows port *fanout_load* values.

To remove *fanout_load* information from ports, use the *remove_fanout_load* command.

Examples

This example sets the fanout_load on ports matching "OUT*" to 3.0.

```
pt_shell> set_fanout_load 3.0 "OUT*"
```

See Also

- remove_fanout_load
- report_constraint
- report_port
- set_max_fanout

---

set_glitch_power_analysis_options

Sets the options for glitch power analysis.

Syntax

Boolean *set_glitch_power_analysis_options*

```
[-exclude_nets exclude_nets]
[-max_fanout max_fanout ]
[-max_slew max_slew]
```

Data Types

```
exclude_nets          list
max_fanout            integer
max_slew              float
cycle_based_glitch    booean
```

Feedback

Arguments

`-exclude_nets` *exclude_nets*

Specifies the net which are selectively disabled for glitch analysis.

`-max_fanout` *max_fanout*

Specifies max_fanout limit for net to be considered as HFN net for glitch analysis.

`-max_slew` *max_slew*

Specifies max_slew limit for net to be considered as HFN net for glitch analysis.

`-cycle_based_glitch`

Enables PrimePower to do clock cycle based glitch detection in time based analysis. This is allowed for regular time_based flow (VCD/FSDB with delays) or delay shifted flow. The power associated with glitch transitions will be categotized as glitch power. The fastest clock period in the design is considered as reference clock period. Depending on the pulse width , a glitch transition will be categorized as inertial glitch (IG) or transport glitch (TG) and this information will be available in the detailed and summary report of 'report_power_calculation'.

Description

Use this command to selectively disable the net during the glitch power analysis.It will impact both the default glith power analysis flow as well as the clock cycle based glitch analysis.

pt_shell> *set_glitch_power_analysis_options -exclude_nets I1 \\ -max_fanout 5 -max_slew 2.0*

See Also

- report_power

---

set_gpd_config

Specifies the parameters for reading parasitic data in GPD format: capacitor filtering thresholds, net subset, and netlist type.

Syntax

string *set_gpd_config*

```
[-absolute_coupling_threshold threshold]
[-relative_coupling_threshold threshold]
[-coupling_threshold_operation and | or]
```

```
[-netlist_select_nets net_names]
[-netlist_type netlist_type_specification]
-gpd gpd_directory
```

Data Types

threshold	float
net_names	list
netlist_type_specification	list
gpd_directory	string

Arguments

`-absolute_coupling_threshold` *threshold*

Specifies the absolute coupling threshold for filtering coupling capacitors. A capacitor having less that this capacitance value is split into two capacitors to ground. The threshold value must be greater than or equal to the corresponding parameter in the original GPD configuration file, COUPLING_ABS_THRESHOLD.

`-relative_coupling_threshold` *threshold*

Specifies the relative coupling threshold for filtering coupling capacitors. A capacitor is split to ground if its capacitance value divided by the total net capacitance is less than this threshold. The threshold value must be greater than or equal to the corresponding parameter in the original GPD configuration file, COUPLING_REL_THRESHOLD.

`-coupling_threshold_operation and | or`

Specifies whether coupling capacitors are filtered when the capacitance value is less than both the absolute AND relative thresholds (the default), or filtered when the value is less than either the absolute OR relative threshold (resulting in more filtering of coupling capacitors). If the original GPD database parameter COUPLING_THRESHOLD_OPERATION is set to OR, you cannot change it.

`-netlist_select_nets` *net_names*

Specifies a subset of nets to be read from the GPD database. The default is to read all nets.

`-netlist_type` *netlist_type_specification*

Specifies the parasitics network form to consider when reading the GPD database, as described the StarRC User Guide under NETLIST_TYPE. The allowed settings are *no_couple*, *R*, *CG*, *CC*, *RCG*, and *RCC* (the default).

`-gpd` *gpd_directory*

Specifies the absolute or relative path to the GPD directory.

Feedback

Description

This command allows you to override certain parameters for reading parasitic data from a Galaxy Parasitic Database (GPD) directory using the *read_parasitics -format gpd* command. The default parameters are defined in a file called the GPD configuration file. This file is generated by the *StarXtract -set_gpd_config* command in the StarRC tool.

The options you specify with the *set_gpd_config* command are appended to the existing GPD configuration file in the GPD directory. Note that this requires write access to the GPD directory.

If there is not already a user-defined GPD configuration file in the GPD directory, the command obtains the configuration values original GPD configuration and writes out a new configuration file.

To report the GPD properties that have been set, use the *report_gpd_config* command.

To cancel the effects of the *set_gpd_config* command and revert to the original GPD database, use the *reset_gpd_config* command.

Examples

The following example sets the absolute and relative thresholds for filtering coupling capacitors in the GPD database called my_design1.gpd.

```
pt_shell> set_gpd_config -gpd my_design1.gpd \\
 -absolute_coupling_threshold 0.005 \\
 -relative_coupling_threshold 0.05
 1
 pt_shell> report_gpd_config -gpd my_design1.gpd
  ...
```

After you run these commands, when you read in this parasitic data with the *read_parasitics* command, a coupling capacitor is split to ground when the capacitance is less than 0.005 and less than 5 percent of the total net capacitance.

See Also

- report_gpd_properties
- report_gpd_config
- reset_gpd_config

---

set_gui_stroke_binding

Set the command binding for a stroke.

Feedback

Syntax

set_gui_stroke_binding

```
dictionary_name
stroke_sequence
[-builtin builtin_cmd_name]
[-clear]
[-tcl_cmd tcl_command]
[-label tcl_command_label]
```

Arguments

dictionary_name

Specify the dictionary to set the binding into. Windows that support stroke commands have one or more dictionaries to use when evaluating stroke commands.

stroke_sequence

Specifies the sequence of grids that defines the path of the stroke. See the *Stroke Sequences* section, below, for more information.

-builtin *builtin_cmd_name*

Specifies the name of a built-in command option to be executed.

-clear

Clears the existing command for the specified stroke sequence.

-tcl_cmd *tcl_command*

Specifies the Tcl command to be executed when the stroke is entered.

-label *tcl_command_label*

Specifies the menu label for the tcl command. This menu is displayed for line and snap_line strokes after a delay to provide information on the current bindings.

Description

Select a stroke by specifying a symbol with the mouse. The location of the down-click starts the stroke, and the path that the mouse follows while the mouse button is pressed determines the stroke that is entered. This stroke is mapped onto a numbered 3x3 grid and is transformed into a series of these grid numbers. A stroke dictionary is used to map this sequence of numbers to the command to be executed. When you release the mouse button, this command is executed.

You can use the *report_gui_stroke_builtins* command to determine the available built-in (non-Tcl) commands. You can use the *report_gui_stroke_bindings* command to determine the existing bindings.

Stroke Types

The stroke command facility supports the following modes for entering strokes: line-based, rectangle-based, and path-based. All stroke entry types generate stroke bindings that are the same.

If you are an occasional user, the line-based and rectangle-based entry modes are easier to specify, but they limit the number of strokes that can be generated. If you are an advanced user, the path-based entry mode, which is more difficult to specify, allows many more strokes to be generated.

See the man page for the *set_gui_stroke_preferences* command for complete information on stroke entry types,

Stroke Sequences

The path of a stroke is mapped onto a 3x3 grid. The size of the grid squares are determined to ensure that the stroke covers the width or height of the grid. The stroke is centered in the grid of horizontal or vertical strokes.

The grid is numbered as shown below:

```
1 2 3
4 5 6
7 8 9
```

The stroke is mapped onto the grid and converted to a sequence of these grid numbers, which is called a *stroke sequence*. For example, a diagonal stroke from the top-left corner of the grid to the bottom-right corner has a stroke sequence of *159*; a left-to-right horizontal stroke has a stroke sequence of *456*.

Since the strokes are path-based, a single grid might occur more than once in a sequence. For example, a stroke that moves horizontally left-to-right and right-to-left has a stroke sequence of *12321*.

Strokes also support the Shift, Control, and Alt modifier keys. If one or more of these modifiers are depressed when the stroke is started, they are applied to the stroke. The modifiers are prepended to the stroke sequence, with *s* delimiting Shift, *c* delimiting control, and *a* specifying Alt. The modifiers are separated from the sequence by using a colon (*:*). For example, a left-to-right horizontal stroke sequence is *123*, if unmodified; *s:123* with Shift; *c:123* with Control; and *sc:123* with Shift-Control modifiers.

A click is supported as a special (degenerate) case of a stroke. If the mouse down-click and mouse up-click event happen close enough together (in both time and location), the stroke is recognized as a click and a stroke sequence of *5* is generated.

The stroke facility also supports the definition of a default command binding that is to be used for any bindings not explicitly set. This default binding is specified by using a simple stroke sequence of *0*. If no binding is registered for a given set of modifiers and sequence,

the tool searches for a stroke binding with the same modifiers and a sequence of *0* and uses that if it is found.

Built-In Commands

The application supporting stroke commands might support operations that are not available from the Tcl command language to be invoked via the stroke command. These commands are given a name, and you can use the *-builtin* option to specify a stroke binding to invoke these commands

The stroke facility always supports the built-in commands for zooming and panning listed below.

Zoom_In

Zoom in by a fixed percentage.

Zoom_Out

Zoom out by a fixed percentage

Zoom_In_Rect

Zoom in to fix the rectangle specified by the bounding box for the stroke.

Zoom_Out_Rect

Zoom out centering on the rectangle specified by the bounding box for the stroke.

Zoom_Full

Zoom and pan to fit the entire drawing centered in the window.

Pan_Center_Rect

Pan to center the rectangle specified by the bounding box for the stroke.

Tcl Command Bindings

The bindings support invoking Tcl-based commands to allow the functionality of the mechanism to be expanded. Tcl-based commands are allowed to contain key values that are substituted with data from the command before they are executed by the interpreter, thus allowing these commands to have access to the context information for the stroke. A leading percent (%) character specifies a key value..

The keys listed below are supported by tcl commands.

%rect

Substitutes the coordinates for the bounding box of the stroke. The value is a list of points {{*x1 y1*} {*x1 y2*}}, where you substitute bounding box coordinates for *x1*, *y1*,*x1*, and *y2*.

%startPoint

Specifies the starting point of the stroke. The value is in the form {*x y*}, where you substitute starting point coordinates for *x* and *y*.

%endPoint

Specifies the ending point of the stroke. The value is in the form {*x y*}, where you substitute starting point coordinates for *x* and *y*.

%sequence

Specifies the stroke sequence that invoked this command.

%modifiers

Specifies the modifiers of the stroke sequence that invoked this command.

Examples

The following example defines a left-to-right horizontal binding to execute the built-in operation that centers the location of the stroke in the viewport.

```
psyn_gui-t> set_gui_stroke_binding Graphics 123 \\
-builtin Pan_Center_Rect
```

The following example calls a tcl command for the Shift-modified, lower-left to upper-right stroke sequence.

```
psyn_gui-t> set_gui_stroke_binding Graphics  \\
-tcl s:753 my_tcl_command
```

The following example calls a tcl command taking a rectangle as an argument for the Shift-control, lower-left to upper-right stroke sequence.

```
psyn_gui-t> set_gui_stroke_binding Graphics \\
ss:753 -tcl {my_tcl_command %rect}
```

See Also

- report_gui_stroke_bindings
- report_gui_stroke_builtins
- set_gui_stroke_preferences

---

set_gui_stroke_preferences

Set preferences controlling stroke command entry.

Feedback

Syntax

set_gui_stroke_preferences

```
[-type stroke_entry_type]
[-shift]
[-ctrl]
[-alt]
[-extended_help_delay ms_delay]
```

Arguments

`-type stroke_entry_type`

Specifies the are *snap_line*, *line*, *rect*, *snap_path*, and *path*.

`-shift`

Sets the type for the shift modifier. This option is valid only when using the *-type* option.

`-alt`

Sets the type for the alt modifier. This option is valid only when using the *-type* option.

`-ctrl`

Sets the type for the ctrl modifier. This option is valid only when using the *-type* option.

`-extended_help_delay ms_delay`

Specifies the number of milliseconds to wait before showing the extended help (menu) for a stroke. A value of less than *500* makes the extended feedback immediate. Extremely large values suppress the extended feedback almost completely.

Description

Select a stroke command by specifying a symbol with the mouse. The location of the down-click starts the stroke, and the path that the mouse follows while the mouse button is pressed determines the stroke that is entered. This stroke is mapped onto a numbered 3x3 grid and is transformed into a series of these grid numbers. A stroke dictionary is used to map this sequence of numbers to the command to be executed. When you release the mouse button, this command is executed.

Use this command to control the user interface behavior when you are specifying a stroke. The preferences allow the strokes to be used by both application experts and occasional users of the application.

The two primary types of stroke entry mechanisms are the line-based type and the stroke type. The primary preference supported is the stroke type. A line-based type determines

the stroke based on the values of the stroke's start and end points. The path between the points is ignored. A line-based stroke turns the stroke facility into a form of "pie-menu" that supports nine different menu items. A path-based type determines the stroke based on the path between the stroke's start and end points. This allows an extremely large number of unique strokes to be defined, and it also requires a significantly-higher amount of precision when specifying the stroke than when using the line-based type.

The snap_line, line, and rect stroke types are all line-based strokes: the snap_path and path are path-based strokes.

Stroke Types

snap_line

The line between the start and end points is snapped onto the nearest 45 degree angle. It allows the generation of eight different stroke sequences and the click stroke sequence.

line

The line between the start and end points specifies the stroke. It allows the generation of eight different stroke sequences and the click stroke sequence.

rect

The rectangle between the start and end points specifies the stroke. It allows the generation of four different stroke sequences and the click stroke sequence.

snap_path

This path-based stroke snaps the segments of the path onto the nearest 45 degree angle to make it easier to generate stroke sequences with diagonals and straight lines. Since it is a path-based stroke, the number of unique sequences supported for this entry type is extremely large.

path

This path-based stroke is a free-formed path from the start point to the end point. Since it is a path-based stroke, the number of unique sequences supported for this entry type is extremely large.

The unmodified stroke type is the default type for all modifiers when a type has not been specified for the modifier. Different stroke types can be used for each combination of modifiers by setting the stroke type for the modifier with this command.

Examples

The following example sets the stroke type to snap_line.

```
psyn_gui-t> set_gui_stroke_preferences -type \\
snap_line
```

Feedback

The following example sets the stroke type for Shift-Control modified strokes to snap_line.

```
psyn_gui-t> set_gui_stroke_preferences -shift  \\
-ctrl -type snap_line
```

See Also

- report_gui_stroke_bindings
- report_gui_stroke_builtins
- set_gui_stroke_binding

---

set_hier_config

Specifies the HyperScale hierarchical analysis configuration options.

Syntax

status *set_hier_config*

```
[-path directory_for_data_storage]
[-block name_of_subblock]
[-instances names_of_instances]
[-name mim_group_name]
[-mim_reference mim_reference_instance_name]
[-enable_mim]
[-enable_top_only_flow]
[-session]
[-top]
[-partitions names_of_instances]
[-split names_of_instances]
[-exclude names_of_instances]
[-affinity host_option_names]
```

Data Types

```
directory_for_data_storage   string
name_of_subblock             string
names_of_instances           list
mim_group_name               string
mim_reference_instance_name  string
host_option_names            list
```

Arguments

`-path directory_for_data_storage`

Specifies the name of the directory in which the HyperScale block data was previously stored by the *write_hier_data* command during block-level timing analysis. Use this option together with the *-block* option to specify a HyperScale block and the location of its timing data.

`-block` *`name_of_subblock`*

Specifies the library name of a HyperScale block that was previously analyzed for timing. Use this option together with the *-path* option to specify a HyperScale block and the directory containing its timing data.

To analyze the timing of the current design, for each instance of the named block, the tool uses the block timing model previously created and stored in the specified directory. When you use the *write_hier_data* command, the tool writes out the context for each HyperScale block existing in the current design.

By default, all instances of the specified block are treated as HyperScale blocks. To specify only a subset of these instances as HyperScale blocks, use the *-instances* option. To specify multiple library blocks as HyperScale blocks, use a separate *set_hier_config* command for each library block.

The *-path* and *-block* options are used in a higher-level discrete analysis session to identify lower-level HyperScale blocks; they are not used in the full-chip automatic distributed analysis flow.

`-instances` *`names_of_instances`*

Specifies a list of one or more instances of the library block specified by the *-block* option. Only the listed instances are treated as HyperScale blocks; they are analyzed by using the HyperScale model previously stored during block-level analysis. Other instances of the same block undergo full-netlist timing analysis without using the HyperScale model. By default, all instances of the block are treated as HyperScale blocks.

For HyperScale context generation, by default, the tool performs context merging for all instances of the block specified by the *-block* option. When you use the *-instances* option, the tool performs context merging only for the listed instances and does not merge the context for instances not in the list. It uses the first instance in the list as the reference and merges the contexts of the other listed instances to match the first instance.

`-name` *`mim_group_name`*

Specifies a name for the multiply instantiated module (MIM) set, consisting of all instances that share the same merged context, whether all instances of the block by default or a subset of those instances by usage of the *-instances* option. Specifying a MIM group name is optional.

Assigning a specific name can be useful if you create multiple MIM groups for instances of the same library block. Later in the HyperScale analysis flow, when you perform block-level analysis, you can load the context of the specific MIM group by using the *-name* option of the *read_context* command.

Multiple subsets of merged instances can share the same MIM group name, and these merged instance groups can be stored in different directories by different *set_hier_config* commands. For block-level analysis, when you use the *read_context* command with the *-name* option, the command reads in the context for all merged instance groups that share the same MIM group name.

`-mim_reference` *`mim_reference_instance_name`*

Specifies the instance name of the multiply instantiated module (MIM) reference used for context merging. By default, MIM context merging automatically chooses a reference instance from the largest mergeable instance cluster. With this option, the specified instance is used for context merging, overriding the default.

`-enable_mim`

Specifies merging of the generated context for multiple HyperScale instances of the same library block in the full-chip distributed analysis flow. By default, the contexts for multiple instances of the same block are not merged in this flow; the context for each individual block is generated separately.

Use this option for more efficient HyperScale block context generation when multiple instances of the same HyperScale block have the same (or very similar) timing environment and therefore share the same timing context.

When the *-enable_mim* option is set, all reports in distributed analysis refer to only the primary MIM partition, except for reports generated by the *report_attribute* command.

`-enable_top_only_flow`

Performs timing analysis of only the top design in the distributed analysis flow, instead of the whole design. In this flow, valid timing reports are available only for active objects (a subset of the top design). To identify active objects, query their is_active attribute.

Use this option for more efficient distributed analysis when you are interested in timing results only for the top design. By default, the compute resources for block partitions are released after a timing update. To change this default behavior, set the *hier_enable_release_resources_in_top_only_flow* variable.

`-session`

Specifies the reading of session data saved by the *save_session* command in the full-chip distributed analysis flow. By default, without the *-session* option, the command reads HyperScale modeling data saved by the *write_hier_data* command.

With the *-session* option, you can read either the top-level session data by using the *-top* option or block session data by using the *-block* option; use the *-path*

option to specify the session data location, whether top or block. Use multiple *set_hier_config* commands with the *-session* option to read in the session data for the top level and for each HyperScale block.

`-top`

Specifies the reading of top-level session data with the *-session* option in the full-chip distributed analysis flow. Use either the *-top* or *-block* option, together with the *-path* option, to specify the top-level or block-level session data to read in.

`-partitions` *`names_of_instances`*

Specifies the block instances to be analyzed separately in the full-chip distributed analysis flow. By default, all hierarchical blocks at the top level are analyzed separately. Using the *-partitions* option overrides the default behavior, causing only the listed blocks at the top level to analyzed separately.

Use this option to specify only the larger blocks for separate analysis, leaving the remaining smaller blocks to be analyzed together with the rest of the top level. This option cannot be used with the *-exclude* or *-split* option.

To assign partitions for analysis on specific host resources, use the *-affinity* option.

`-split` *`names_of_instances`*

Specifies the blocks contained at the top level whose next-lower-level subblocks are to be analyzed separately in the full-chip distributed analysis flow. By default, all hierarchical blocks at the top level are analyzed separately and all blocks at levels below the first level are not analyzed separately.

Using *-split* option extends the splitting of the design into smaller units to two levels down instead of only one level down, for the specified instances. Use this option to break down the analysis of very large blocks into smaller units.

For each named instance, HyperScale distributed analysis performs a separate analysis of each highest-level subblock in that instance. The rest of the logic in that instance is pulled up to the top level and analyzed as part of the top level.

`-exclude` *`names_of_instances`*

Specifies the block instances at the top level to be excluded from separate analysis in the full-chip distributed analysis flow, overriding the default behavior. The listed instances are analyzed together with the top level instead of separately. Use this option to keep small blocks with the top level.

`-affinity` *`host_option_names`*

Defines the resource affinity for the partitions listed in the *-partitions* option. The *-affinity* option specifies a list of host options previously defined by

the *-name* option of the *set_host_options* command. The tool assigns the partitions for analysis on the worker processes defined by the corresponding *set_host_options* commands.

You can use the *-affinity* option only with the *-partitions* option, and only after the design has been linked.

If you specify multiple host option names, the tool assigns the partitions with the largest number of cells to the resources that have the largest reserved memory. To assign each partition to a specific host resource, use a separate *set_hier_config* command for each partition-to-resource assignment.

Description

The *set_hier_config* command specifies the HyperScale hierarchical analysis configuration options. You must set this configuration before you link the design, and HyperScale analysis must be enabled by setting the *hier_enable_analysis* variable to *true*.

For top-level discrete HyperScale analysis, use the following *set_hier_config* options to specify each HyperScale block and the block's associated data storage directory:

```
set_hier_config
  -path directory_for_data_storage
  -block name_of_subblock
  [-instances names_of_instances]
  [-name mim_group_name]
```

For full-chip distributed analysis, use the following *set_hier_config* options to enable merging of multiply instantiated blocks, to load existing session data, or to override the default block partitioning:

```
set_hier_config
  [-enable_mim]
  [-session  -top | -block name_of_subblock -path session_dir]
  [-partitions names_of_instances | -exclude names_of_instances]
  [-split names_of_instances]
```

HyperScale Technology

HyperScale technology greatly increases timing analysis performance and capacity by controlling the visibility into the design hierarchy. With accurate data reduction and automated design visibility control, and with the block-level timing data saved in the database, a combined block-level and top-level timing analysis requires significantly less memory and takes significantly less runtime.

In the HyperScale flow, the visibility at the top level is the portion of design that consists of the interblock glue logic and the block interfaces connected at the top level. This is the only portion of the design that the top-level timing integration actually needs for accurate analysis.

HyperScale analysis flow makes no sacrifices in the quality of results. It is designed to provide a level of accuracy that is identical or equivalent to that achieved by classic flat analysis where full visibility of the entire design is available from the top level.

The tool offers both discrete and full-chip distributed HyperScale analysis flows. In the discrete flow, you analyze each HyperScale block and the top level in a separate tool session. In that case, you use *set_hier_config* commands to specify which blocks at the top level are HyperScale blocks and the name of each block's storage directory containing modeling and context data. When you use multiple *set_hier_config* commands, a more specific setting overrides a more general one, and a new setting overrides any earlier conflicting setting.

In the full-chip distributed analysis flow, the tool automatically distributes the analysis of HyperScale blocks to a specified set of hosts. In that case, to override the default distribution of blocks to analysis processes, use the *set_hier_config* command to specify which blocks to analyze separately. You can optionally specify the usage of existing top-level or block-level analysis session data previously saved with the *save_session* command instead of data saved by the *write_hier_data* command. You can also specify whether to merge the context data for multiply instantiated modules (MIMs).

Discrete HyperScale Analysis

In the discrete HyperScale analysis flow, you use a separate PrimeTime session for each HyperScale block and for the top level.

For block-level analysis, use the *read_context* command to read the block context generated by top-level analysis, or use budgeted constraints if the block context information is not available; and upon completion of timing analysis, use the *write_hier_data* command to write out the block-level modeling data for future top-level analysis.

For top-level analysis, use the *set_hier_config* command to specify the blocks to be handled as HyperScale blocks. Upon completion of timing analysis, use the *write_hier_data* command to write out the block-level context for future block-level analysis. You can iterate between block-level and top-level analysis to handle any design or constraint changes.

In the top-level analysis session, use the following *set_hier_config* command options to specify each HyperScale block and the block's associated data storage directory:

```
set_hier_config
  -path directory_for_data_storage
  -block name_of_subblock
  [-instances names_of_instances]
  [-name mim_group_name]
```

Using the *-block* option, specify the library block name, and using the *-path* option, specify the path to the directory used for block model and context data storage. If multiple instances of that block exist in the design, by default they are all treated as HyperScale

blocks and share the same merged context. To specify only a subset of these instances as HyperScale blocks, list those instances with the *-instances* option. Use a separate *set_hier_config* command for each HyperScale library block.

You can optionally specify a name for the multiply instantiated module (MIM) set, consisting of all instances that share the same merged context. This can be useful if you create multiple MIM groups for instances of the same library block. Later in the HyperScale analysis flow, when you perform block-level analysis in a discrete tool session, you can load the context of the specific MIM group by using the *-name* option of the *read_context* command.

Full-Chip Distributed Analysis

In the full-chip distributed analysis flow, you specify a full-chip analysis in a single PrimeTime session, and the tool automatically distributes analysis of blocks to separate processes on different hosts. For example:

```
# Enable hierarchical analysis
set_app_var hier_enable_analysis true

# Set working directory for analysis data
set_app_var hier_distributed_working_directory $nfs_dir

# Specify hosts for automated distributed analysis
set_host_options -num_processes $nProc1 -max_cores $mCore1
 -submit_command ...
set_host_options -num_processes $nProc2 -max_cores $mCore2
 -submit_command ...
 ...
start_hosts

# Run full-chip flat analysis script
read_verilog blk_A.v
read_verilog blk_B.v
read_verilog top.v
link_design TOP
 ...
update_timing
 ...
```

Use the following *set_hier_config* command options to enable context merging for multiply instantiated modules (MIMs), to load existing top-level or block-level analysis session data for the current full-chip analysis, or to override the default block partitioning:

```
set_hier_config
  [-enable_mim]
  [-session  -top | -block name_of_subblock -path session_dir]
  [-partitions names_of_instances | -exclude names_of_instances]
  [-split names_of_instances]
```

In the full-chip distributed analysis flow, by default the context is *not* merged for MIMs. To enable context merging for MIMs, use the *-enable_mim* option.

To load existing top-level or block-level analysis session data (saved by the *save_session* command) for the current full-chip analysis, use the *-session* and *-top* options or the *-session*, *-block*, and *-path* options, respectively. Use a separate *set_hier_config* command for the top level and for each HyperScale library block. By default, without the *-session* option, the command reads HyperScale modeling data saved by the *write_hier_data* command.

By default, the distributed analysis flow treats all hierarchical blocks at the top level as HyperScale blocks and analyzes each such block using a single separate process. To override the default block distribution, use the *-partitions*, *-split*, and *-exclude* options:

- The *-partitions* option causes only the listed blocks to analyzed separately, leaving the unlisted blocks to be analyzed with the top level.
- The *-exclude* option does the opposite; it causes the listed blocks to be analyzed with the top level, allowing the unlisted blocks to be analyzed separately.
- The *-split* option breaks down the listed blocks even further, allowing their highest-level subblocks to be analyzed separately.

Top-Down and Bottom-Up Views

From a bottom-up point of view, HyperScale timing analysis typically starts with the block-level designers implementing each block with prescribed budgeting constraints, and performing timing analysis and optimization to ensure the block-internal paths meet the timing requirements. Then they use the *write_hier_data* command to store all block timing data into the HyperScale database.

After that, at the top level of hierarchy (or at the next-higher, intermediate level of hierarchy), the tool reads the block-level timing model data from the HyperScale database for timing analysis. For the case of an intermediate hierarchy level, it can generate its own timing analysis data and pass the timing model further up to its parent-level hierarchy, again using the *write_hier_data* command.

From a top-down point of view, HyperScale timing analysis uses the already-saved block models to perform the timing analysis for the visible portion of the design, including paths at the top level and paths that cross the boundaries of the HyperScale blocks, but excluding register-to-register paths inside the HyperScale blocks. This top-level analysis also checks the actual timing and environmental context at the subblock boundaries for all instances of HyperScale blocks to ensure that the block contexts match the conditions used during block-level analysis. To report the details boundary checking, use the *report_constraint* command.

At the top level, if further refinement of the subblocks is necessary, you can save the block context by using the *write_hier_data* command. Then, for subsequent block-level analysis,

the tool uses the stored context information for accurate refinement and optimization of the block in the context of the higher level.

In a top-down analysis flow in which the lower-level block netlists are not yet available, you can generate the block-level constraints from the top level by using the *characterize_context* and *write_context* commands.

Examples

To convert a full-chip flat analysis script to a HyperScale hierarchical analysis script, add lines at the beginning of the script similar to the following:

```
set_app_var hier_enable_analysis true
set_app_var hier_distributed_working_directory $nfs_dir
set_host_options -num_processes $nProc1 -max_cores $mCore1
 -submit_command ...
set_host_options -num_processes $nProc2 -max_cores $mCore2
 -submit_command ...
start_hosts
```

These lines invoke HyperScale automated distributed analysis, which treats all hierarchical blocks at the top level as HyperScale blocks. It analyzes each of these blocks using a separate process. To override the default behavior, use the *set_hier_config* command. For example,

```
set_hier_config \\
 -enable_mim \\       # Merge context for MIM blocks
 -exclude {U1} \\     # Keep block instance U1 with top level
 -split {U6}         # Split block instance U6 further
```

To perform a discrete analysis for each HyperScale block and for the top level, use commands similar to the following.

```
# Top-Level Analysis Session
set hier_enable_analysis true
 ...
set_hier_config -block BLKA -path $model_A
set_hier_config -block BLKB -path $model_B
 ...
read_verilog TOP.v
link_design TOP
 ...
update_timing
 ...
write_hier_data $topContextDir

# Block BLKA Analysis Session
set hier_enable_analysis true
 ...
read_verilog BLKA.v
link_design BLKA
 ...
```

```
read_context $topContextDir
 ...
update_timing
 ...
write_hier_data $model_A

# Block BLKB Analysis Session
set hier_enable_analysis true
 ...
read_verilog BLKB.v
link_design BLKB
 ...
read_context $topContextDir
 ...
update_timing
 ...
write_hier_data $model_B
```

You can iterate between the top-level and block-level analysis sessions to close timing or to handle design changes.

For each initial block-level analysis, you can use budgeted constraints, or you can generate block-level constraints from the top level by the *characterize_context* and *write_context* commands:

```
# Perform full-chip extraction of block constraints
# from top level (minimal update, no parasitics)
set_app_var hier_characterize_context_mode constraints_only
 ... read, link ...
characterize_context -block blkA
update_timing
write_context -format ptsh blkA -output $cnsDir

# Perform initial block-level analysis using extracted constraints
source $cnsDir/CONSTRAINT/blkA/.../variables.pt
source $cnsDir/CONSTRAINT/blkA/.../constraints.pt
 ... read, link, update timing ...
write_hier_data $blkDir1
```

In the following example, the design has three levels of hierarchy: TOP, MID, and BOT. Starting from lowest level, the following commands configure the analysis.

```
# BOT Analysis Script
set hier_enable_analysis true
read_context /design/blocks/MID
 ...
write_hier_data /design/blocks/BOT
```

For MID, the tool loads the timing data saved for its subblock BOT and the timing context pushed down from the parent-level design TOP:

```
# MID Analysis Script
set hier_enable_analysis true
read_context /design/TOP
set_hier_config -path /design/blocks/BOT -block BOT
 ...
write_hier_data /design/blocks/MID
```

For TOP, the tool loads the timing data saved for its subblock MID:

```
# TOP Analysis Script
set hier_enable_analysis true
set_hier_config -block MID -path /design/blocks/MID
 ...
write_hier_data /design/TOP
```

This HyperScale setup can significantly improve the runtime and memory for analysis of TOP.

The following commands define the host resources and analyze block instances P1, P2, and P3 as HyperScale blocks using the specified resources. All other hierarchical blocks at the top level are analyzed together with the top level.

```
set_host_options -name RA -num_processes 1 -max_cores 2 ... host1
set_host_options -name RB -num_processes 1 -max_cores 1 ... host2
set_host_options -name RC -num_processes 1 -max_cores 1 ... host3
start_hosts
report_host_usage -verbose
 ...
set_hier_config -partitions {P1 P2} -affinity {RA RB} ...
set_hier_config -partitions {P3} -affinity {RC} ...
 ...
```

The tool assigns partitions P1 and P2 for analysis on resources RA and RB, and partition P3 for analysis on resource RC. The larger of partition P1 or P2 (based on number of cells) is analyzed on the larger of resources RA and RB (based on allocated memory).

See Also

- link_design
- read_context
- report_constraint
- save_session
- set_host_options
- write_hier_data

Feedback

- hier_enable_analysis
- hier_enable_release_resources_in_top_only_flow

---

set_hier_resource_limits

Specifies the constraints on resources to assist in faster runtime of Distributed Hyperscale analysis.

Syntax

status *set_hier_resource_limits*

```
[-max_mem_per_block max_mem_per_block]
[-max_cores_per_block max_cores_per_block]
[-total_cores total_cores]
```

Data Types

```
max_mem_per_block           string
max_cores_per_block         string
total_cores                 string
```

Arguments

`-max_mem_per_block max_mem_per_block`

Specifies max memory in megabytes that can be allocated per hyperscale block. Value should be greater than 0.

`-max_cores_per_block max_cores_per_block`

Specifies max cores that can be allocated per hyperscale block. Value should be greater than 0.

`-total_cores total_cores`

Specifies total core budget that we have for assigning to Distributed hyperscale. Value should be greater than 0.

This commands setups constraints for optimal partitioning and resource allocation that is reported by *report_hier_analysis command with suboption -explore_partitions*

Feedback

Examples

Given a design with multiple hyperscale blocks. To allocate 14 processors where we can maximum of 4 processors that can be allocated for each hyperscale block and with 16G per block max memory that can be allocated we specify constraints :

set_hier_resource_limits -max_memory_per_block 16000 -total_cores 14 -max_cores_per_block 4

We then invoke partitioner to list out suggestions that should be done to distribute load evenly:

report_hier_analysis -explore_partitions

See Also

- report_hier_analysis
- reset_hier_resource_limits

---

set_host_options

Specifies the host options for computation resources used in threaded, parallel, multi-scenario, and HyperScale analysis.

Syntax

status *set_host_options*

```
[-max_cores number]
[-local_process]
[-num_processes number]
[-load_factor number]
[-name name]
[-submit_command command]
[-terminate_command command]
[-protocol auto | custom | lsf | netbatch | pbs | rsh | rtda | sge | sh |
 slurm | ssh]
[-reserved_memory number]
[host_names]
```

Data Types

```
number      integer
name        string
command     string
host_names  list
```

Feedback

Arguments

`-max_cores` *number*

Specifies the maximum number of CPU cores that can be used for the current local process, or for each subsequently launched remote process if the *-num_processses* option is used. The number can be an integer between 1 and the number of cores available on the host, up to a maximum of 260. Setting a number larger than 16 requires additional licenses to be checked out; see the *PrimeTime User Guide* for details. The default maximum is 4 cores per host.

`-local_process`

Sets the resource usage limits for the current local process, overriding the default limits. The *-local_process* and *-num_processes* options are mutually exclusive. If neither of these options is used, the command applies to the local process by default.

`-num_processes` *number*

Specifies the number of processes to launch per host for future PrimeTime jobs and creates a set of host options for those processes. You can use multiple *set_host_options -num_processes* commands to create multiple sets of processes to launch, each with its own host options, and optionally assign a name to each set by using the *-name* option. When the *-num_processes* option is used, the command does not affect the current local process.

`-load_factor` *number*

Specifies the effective load per host, either 1 (the default) or 2. When the number of scenarios exceeds the number of available hosts, setting a larger load factor reduces or eliminates session save and restore operations, saving time at the cost of more memory.

Each real worker process (specified by *-num_processes*) can in turn launch virtual workers on the same host, such that the total number of workers (real and virtual) equals the specified load factor. The real and virtual workers take turns using cores (specified by *-max_cores*) of the host. Because virtual workers are processes forked by the real workers, they do not consume slots on the farm.

In the context of DMSA, the virtual workers are identical PrimeTime sessions that run scenarios. This option is useful when the number of hosts available on the farm is less than the number of scenarios. By launching additional virtual workers, multiple scenarios are held in memory, thereby reducing the number of automatic save restore operations that would otherwise consume a lot of runtime. Note that storing multiple scenarios in memory increases the total memory usage by approximately a factor of *-load_factor*. Account for this additional usage when you reserve memory on the farm.

`-name` *`name`*

Specifies a name for the set of host options being defined with the *-num_processes* option, allowing you to reference that set of options in future commands. If the name already exists, the new host option settings overwrite the existing settings, and any remote processes already started under that name are shut down.

If you do not use the *-name* option, the set of host options created by the *-num_processes* option is automatically assigned a new name: AUTO_0, AUTO_1, and so on. Only explicitly (not automatically) named option sets can be used with the *-affinity* option of the *create_scenario* command.

`-submit_command` *`command`*

Specifies the command that the tool uses to submit and launch PrimeTime processes associated with the *-num_processes* option. The specified submit command should be non-interactive in nature. When the command is executed, the tool captures the job/process ID, which you can report by using the *report_host_usage* command.

By default, if the *-submit_command* option is not used, and if the *host_name* option is not used (or it specifies the same host as the local host or the host of the manager process), the processes are launched on the local host or by the operating system of the manager; or if the specified host is a different host, the tool uses the *rsh* command by default to make remote connections to the host for setting up the processes.

`-terminate_command` *`command`*

Specifies the command that the tool uses to terminate jobs submitted by the *-submit_command* option (for example, upon exit).

`-protocol auto | custom | lsf | netbatch | pbs | rsh | rtda | sge | sh | slurm | ssh`

Specifies the type of the communication protocol used for submitting jobs. The default is *auto*, which automatically infers the protocol from the command specified in the *-submit_command* option. For automatic selection, the base name of the submission command must match a known protocol submission command: sh, rsh, ssh, bsub (LSF), qsub (SGE), nc (RTDA), qsub (PBS), or sbatch (SLURM). If no match is found, the tool infers a custom protocol. For more information, see "Establishing the Protocol" in the DESCRIPTION section.

`host_names`

Specifies a list of names of hosts on which to launch worker processes, for the set of processes created by the *-num_processes* option. By default, if you do not specify a list of host names, the command applies to the local host on which

the current process is running, and the *report_host_usage* command reports the host name as "localhost".

If any host names are specified other than localhost or the name of the host on which the manager process is running, you can use the *-submit_command* option to specify the command that the tool uses to set up a remote connection to the hosts to launch PrimeTime worker processes.

Note: If a custom protocol is used to directly target a specific host, the host must be part of the *-submit_command* string; it cannot be specified as the *host_names* option, because the PrimeTime tool cannot understand the relationship between the submit command and how that command targets a host.

`-reserved_memory` *number*

Specifies the amount of memory reserved for each process associated with the *-num_processes* option. Ensure that the value matches the memory reserved by the *-submit_command* option. The number can be an integer between 1 and 2147483646, representing the number of Gigabytes of memory reserved for each process. The *report_host_usage* command generates a reserved memory section in its report only if the *-reserved_memory* option is used.

Description

This command specifies the host options for the current process or for launching future PrimeTime processes. The PrimeTime tool uses multiple processes for threaded, parallel, multi-scenario, and HyperScale analysis.

To specify the maximum number of cores to use on the current local host, use the *-max_cores* option. For example,

```
pt_shell> set_host_options -max_cores 6
Information: The max_cores limit for the local (current) process
 has been modified by 2 to 6. (CMCR-103)
```

The default maximum is 4, or the number of physically available cores if that number is less than 4.

To specify the host options for launching future processes, use the *-num_processes* option together with the desired options:

```
set_host_options
  -num_processes number                  ; processes to launch per host
  [host_names]                           ; default is localhost
  [-max_cores number]                    ; default is 4
  [-submit_command command]              ; example: "/usr/bin/rsh"
  [-terminate_command command]           ; example: "[sh which bkill]"
  [-protocol auto | custom | protocol_name]    ; example: lsf
  [-name name_of_host_option_set]        ; default AUTO_0, AUTO_1, ...
```

```
  [-reserved_memory number]                ; reserved memory (GB)
preprocess
```

To report the host option settings, use the *report_host_usage* command.

Specifying these host options does not actually launch the processes. For details, see the man page for the *start_hosts* command.

You can use multiple *set_host_options -num_processes* commands to create multiple sets of processes to launch, each with its own host options, and optionally assign a name to each set by using the *-name* option. Then you can reference each set of options by name.

By default, the tool launches remote processes by using the pt_shell executable "$synopsys_root/bin/pt_shell". To specify a different executable or a custom wrapper script for launching remote processes, use the *set_distributed_parameters* command.

Establishing the Protocol

The *-protocol* option specifies the type of the communication protocol used for submitting jobs:

```
-protocol auto | custom | lsf | netbatch | pbs | rsh | rtda | sge | sh |
 slurm | ssh
```

The following rules apply to usage of the *-protocol* option and related usage of the *-submit_command* option:

- The PBS and SGE protocols use the same submission command (qsub). By default, automatic selection assumes that the protocol is SGE. To use PBS instead, explicitly specify *-protocol pbs*.
- A wrapper script using sh, rsh, or ssh should explicitly specify the protocol, for example, using *-protocol sh*.
- To target a protocol that is not one of the of known protocols, or if you must wrap a known protocol with a script that alters the protocol submit command interface, then use the *-protocol custom* option with the *-submit_command* option. In that case, the *-submit_command* option should specify the custom submit script to execute (along with appropriate options) to submit a worker job.
- If you wrap a known protocol with a script that does NOT alter the protocol submit command interface, you can select the underlying protocol by using the *-protocol* option. In that case, the *-submit_command* should specify a custom submit script that passes all options accepted by the protocol through to the underlying protocol submit command, as the PrimeTime tool passes additional arguments through to the protocol submit command. If the base name of the script does not match the normal protocol submit command name, a warning message tells you to ensure compatibility between the submit command and the selected protocol.

- If the custom protocol is used to directly target a specific host, the host must be part of the *-submit_command* string; it cannot be specified in the *host_names* option.
- The command argument of the *-submit_comamnd* option cannot contain the semicolon character (;). Using that character triggers an immediate error when you attempt to run the *set_host_options* command.
- If you use *-protocol sge*, you cannot pass "-b y" in the *-submit_command* argument. Doing so triggers an immediate error (or a launch failure if "-b y" is inside a qsub call enclosed in a wrapper).
- If you use *-protocol rtda*, you cannot pass not pass "-wl" or "-w" in the *-submit_command* argument. Doing so triggers an immediate error (or a launch failure if "-wl" or "-w" is inside an nc call enclosed in a wrapper).
- If you use *-protocol lsf*, you cannot pass not pass "-K" in the *-submit_command* argument. Doing so triggers an immediate error (or a launch failure if "-K" is inside a bsub call enclosed in a wrapper).
- If you use PrimeTime inside the container environment initiated by launching pt_shell with the *-container* option or by setting the *SNPS_CONTAINER* environmental variable, you cannot use protocols *rsh* or *custom*.

Examples

In the following examples, the manager process is running on the host named roger15. Multiple *set_host_options* commands specify several different sets of host options.

The following command specifies the execution of one process on the same host as the manager, and names this set of host options "my_opts1". It does not specify a limit on the maximum number of cores to use, so the default value of 4 applies (or the number of available cores, if less than 4).

```
pt_shell> set_host_options -name my_opts1 -num_processes 1
```

The following command does the same as the first example, but explicitly states the name of the host and limits the maximum number of cores to 2, and names this set of host options "my_opts2".

```
pt_shell> set_host_options -name my_opts2 -num_processes 1 \
          roger15 -max_cores 2
```

The following command specifies the usage of one process on the host named roger11, sets the maximum number of cores to 3, and names this set of host options "my_opts3". Because the command does not specify a submit command, "rsh" will be used to connect to roger11, as indicated in a warning message.

```
pt_shell> set_host_options -name my_opts3 -num_processes 1 \
          roger11 -max_cores 3
```

```
Warning: no submit command specified to access remote host
 'roger11'. The 'rsh' command will be used. (CMCR-004)
```

The following command specifies the usage of one process on the host named roger12 using the submit command "/usr/bin/rsh", and names this set of host options "my_opts4". The default maximum number of cores applies.

```
pt_shell> set_host_options -name my_opts4 -num_processes 1 \
          -submit_command "/usr/bin/rsh" roger12
```

The following command explicitly specifies the type of the communication protocol to be used (lsf), using the "bsub" command found from the environment path (sh which bsub) to submit the job, requiring 1 GB of memory and 2 slots per compute server, using a maximum of 2 cores. The farm job must be specified with the number of slots needed to support the number of cores to use with the -n 2 -R span[ptile=2] options; the square brace characters must be escaped. To terminate jobs that do not come online, the tool uses the "bkill" command found from the environment path. The name of this set of options is "my_opts5".

```
pt_shell> set_host_options -name my_opts5 -num_processes 1 \
          -protocol lsf \
          -submit_command "[sh which bsub] -n 2 -R rusage\[mem=1000\] \
          -R span\[ptile=2\]" -terminate_command "[sh which bkill]" \
          -max_cores 2
```

The following command specifies the usage of 2 processes on a Grid farm, using the "qsub" command found from the environment path to submit the job, using the project named "bnormal", requiring 1 GB of memory and 4 slots per compute server, using a maximum of 4 cores. The farm job must be specified with the number of slots needed to support the number of cores to use by targeting the parallel environment "mt", requesting 4 slots with the "-pe mt 4" options. To terminate jobs that do not come online, the tool uses the "qdel" command found from the environment path. The name of this set of options is "my_opts6".

```
pt_shell> set_host_options -name my_opts6 -num_processes 2 \
          -submit_command "[sh which qsub] -P bnormal -pe mt 4 \
          -l mem_free=1G" -terminate_command "[sh which qdel]" \
          -max_cores 4
1
```

To report the host options created by the *set_host_options* command, run the *report_host_usage* command with the *-verbose* option before starting the hosts.

Note: The submit command for my_opts1 and my_opts2 are empty because the processes are directly launched on the manager's host.

```
pt_shell> report_host_usage -verbose
 ...

  Options Name    Host Name    Num Processes          Protocol
```

Feedback

```
 ------------------------------------------------------------
 my_opts1        roger15      1                      auto
 my_opts2        roger15      1                      auto
 my_opts3        roger11      1                      auto
 my_opts4        roger12      1                      auto
 my_opts5        >>farm<<     1                      lsf
 my_opts6        >>farm<<     2                      auto

 Options Name    Host Name    Action     Command
 ------------------------------------------------------------
 my_opts1        roger15      SUBMIT     <execute directly on manager's
host>
 my_opts2        roger15      SUBMIT     <execute directly on manager's
host>
 my_opts3        roger11      SUBMIT     rsh
 my_opts4        roger12      SUBMIT     /usr/bin/rsh
 my_opts5        >>farm<<     SUBMIT     /lsf/bin/bsub -n 2 -R
rusage[mem=1000]
                                           -R span[ptile=2]
                             TERMINATE  /lsf/bin/bkill
 my_opts6        >>farm<<     SUBMIT
 /remote/sge2/default/bin/lx-amd64/qsub
                                           -P bnormal -pe mt 4 -l
mem_free=1G

-o /abc/work_tests/dmsa_tests/sge.out

-e /abc/work_tests/dmsa_tests/sge.err
                             TERMINATE
 /remote/sge2/default/bin/lx-amd64/qdel

 Usage limits (cores)

 Options Name     #    User-set  Hardware      Effective
 ------------------------------------------------------------------
 (local process)  -    4         8             4
 my_opts1              4         -             -
 my_opts2              2         -             -
 my_opts3              3         -             -
 my_opts4              4         -             -
 my_opts5              2         -             -
 my_opts6              4         -             -
 ------------------------------------------------------------------
 Total                 4

 Memory usage

 Options Name     #                          Memory (MB)
 ------------------------------------------------------------------
 (local process)  -                          236.41

 Performance
```

```
 Options Name      #    CPU Time (s)           Elapsed Time (s)
 ------------------------------------------------------------------
 (local process)   -    2                      3
```

Use the *start_hosts* command to launch hosts specified in all host options, then use the *report_host_usage* command with the *-verbose* option to see the current status of remote hosts. The protocol column in the updated report shows the name of protocol used for a particular host option. For managed resources, the Job ID column shows the number that can be used to find the job on a farm.

```
pt_shell> report_host_usage -verbose
 ...

 Options Name       Host Name    Num Processes            Protocol
 ------------------------------------------------------------------
 my_opts1           roger15      1                        sh
 my_opts2           roger15      1                        rsh
 my_opts3           roger11      1                        rsh
 my_opts4           roger12      1                        rsh
 my_opts5           >>farm<<     1                        lsf
 my_opts6           >>farm<<     2                        sge

 Options Name  #    Host Name    Job ID    Process ID    Status
 ------------------------------------------------------------------
 my_opts1      1    roger15      -         9308          ONLINE
 my_opts2      2    roger15      -         9362          ONLINE
 my_opts3      3    roger11      -         20818         ONLINE
 my_opts4      4    roger12      -         29080         ONLINE
 my_opts5      5    madyak633    373702    26253         ONLINE
 my_opts6      6    peemt801     2184617   17287         ONLINE
               7    peemt801     2184618   17288         ONLINE

 Options Name       Host Name    Action     Command
 ------------------------------------------------------------
 my_opts1           roger15      SUBMIT     <execute directly on
manager's host>
 my_opts2           roger15      SUBMIT     <execute directly on
manager's host>
 my_opts3           roger11      SUBMIT     rsh
 my_opts4           roger12      SUBMIT     /usr/bin/rsh
 my_opts5           >>farm<<     SUBMIT     /lsf/bin/bsub -n 2 -R
rusage[mem=1000]
                                             -R span[ptile=2]
                                TERMINATE  /lsf/bin/bkill
 my_opts6           >>farm<<     SUBMIT
 /remote/sge2/default/bin/lx-amd64/qsub
                                             -P bnormal -pe mt 4 -l
mem_free=1G

-o /abc/work_tests/dmsa_tests/sge.out

-e /abc/work_tests/dmsa_tests/sge.err
```

```
                                  TERMINATE
/remote/sge2/default/bin/lx-amd64/qdel

Usage limits (cores)

Options Name     #     User-set  Hardware      Effective
-------------------------------------------------------------
(local process)  -     4         8             4
my_opts1         1     4         8             -
my_opts2         2     2         8             -
my_opts3         3     3         8             -
my_opts4         4     4         8             -
my_opts5         5     2         12            -
my_opts6         6     4         16            -
                 7     4         16            -
-------------------------------------------------------------
Total                                          4

Memory usage

Options Name     #               Memory (MB)
-----------------------------------------------------
(local process)  -               418.35
my_opts1         1               267.41
my_opts2         2               267.04
my_opts3         3               269.54
my_opts4         4               277.53
my_opts5         5               545.97
my_opts6         6               352.44
                 7               352.82

Performance

Options Name     #    CPU Time (s)   Elapsed Time (s)
-----------------------------------------------------
(local process)  -    2              33
my_opts1         1    2              2
my_opts2         2    2              2
my_opts3         3    2              3
my_opts4         4    2              3
my_opts5         5    2              2
my_opts6         6    2              2
                 7    2              2
```

The following example launches the PrimeTime tool on a 16-core machine. Initially, reporting host usage shows that the local process is effectively limited to 4 cores, the default. Subsequently, you raise the cores limit to 16 by using the *set_host_options* command.

```
pt_shell> report_host_usage -verbose
  ...
```

```
  Usage limits (cores)

  Options Name     #                 User-set  Hardware      Effective
  ------------------------------------------------------------------
  (local process)  -                 4         16            4
  ------------------------------------------------------------------
  Total                                                      4

  Memory usage

  Options Name     #                           Memory (MB)
  ------------------------------------------------------------------
  (local process)  -                           287.65

  Performance

  Options Name     #    CPU Time (s)           Elapsed Time (s)
  ------------------------------------------------------------------
  (local process)  -    2                      6

1

pt_shell> set_host_options -max_cores 16
Information: The max_cores limit for the local (current) process has been
 modified
  by 12 to 16. (CMCR-103)
1
pt_shell> report_host_usage -verbose
 ...

  Usage limits (cores)

  Options Name     #                 User-set  Hardware      Effective
  ------------------------------------------------------------------
  (local process)  -                 16        16            16
  ------------------------------------------------------------------
  Total                                                      16

  Memory usage

  Options Name     #                           Memory (MB)
  ------------------------------------------------------------------
  (local process)  -                           364.57

  Performance

  Options Name     #    CPU Time (s)           Elapsed Time (s)
  ------------------------------------------------------------------
  (local process)  -    2                      85

1
```

Feedback

The following example creates sets of host options with and without virtual workers using the *set_host_options* command, and launches them.

```
pt_shell> set_host_options -num_processes 3
1
pt_shell> report_host_usage -verbose
 ...

  Options Name          Host Name    Num Processes           Protocol
  ------------------------------------------------------------------
  AUTO_0                localhost    3                       auto

 ...
1
pt_shell> set_host_options -num_processes 2 -load_factor 2
1
pt_shell> report_host_usage
 ...

  Options Name          Host Name    Num Processes           Protocol
  ------------------------------------------------------------------
  AUTO_0                localhost    3                       auto
  AUTO_1                localhost    4                       auto

 ...
1
pt_shell> start_hosts
Launching 5 Distributed Worker(s)
which can additionally launch 2 virtual worker(s)
 ...

 ------------------------------------------------------------------------
---
  Distributed farm creation timeout   : 21600 seconds
  Waiting for   5 (of   5) workers    : Wed Apr 25 08:17:25 2018
  Waiting for   0 (of   5) workers    : Wed Apr 25 08:17:35 2018

 ------------------------------------------------------------------------
---

1
pt_shell> report_host_usage

 ...

  Options Name          Host Name    Num Processes           Protocol
  ------------------------------------------------------------------
  AUTO_0                localhost    3                       auto
  AUTO_1                localhost    4                       auto

  Options Name     #    Host Name    Job ID    Process ID    Status
  ------------------------------------------------------------------
```

```
  AUTO_0           1    localhost    -         16536         ONLINE
                   2    localhost    -         16535         ONLINE
                   3    localhost    -         16537         ONLINE
  AUTO_1           4    localhost    -         16625         ONLINE
                   5    localhost    -         16843         ONLINE
                   6    localhost    -         16630         ONLINE
                   7    localhost    -         16812         ONLINE

 ...
1
```

See Also

- create_scenario
- parallel_execute
- remove_host_options
- report_host_usage
- set_distributed_parameters
- set_hier_config
- start_hosts
- stop_hosts
- multi_core_allow_overthreading

---

set_ideal_aggressor

Marks the specified aggressor nets as ideal aggressors. Sets the *si_ideal_aggressor* attribute on these nets.

Syntax

status *set_ideal_aggressor*

```
        net_list
```

Data Types

net_list list

Arguments

net_list

Specifies a list of aggressor nets.

Description

This command marks the specified aggressor nets as ideal aggressor nets and sets the *si_ideal_aggressor* attribute these nets. Ideal aggressor nets are handled specially when they are agressing other nets for the purpose of crosstalk delay and noise calculations. The secondary coupling of these nets is ignored for SI computations.

For regular aggressor nets, secondary coupling to other nets is treated as ground capacitance for SI computations. For special aggressor nets coupled with a large number of other nets, the secondary coupling can be very large and the effective crosstalk impact of these nets may be mitigated due to this. These nets can be marked as ideal aggressors for their secondary coupling to be ignored.

You can query the *si_ideal_aggressor* attribute by using the *get_attribute* command. To remove the ideal aggressor marking on a net, use the *remove_ideal_aggressor* command.

Examples

The following command marks aggressor nets "n_ideal_aggr1" and "n_ideal_aggr2" as ideal aggressor nets.

```
pt_shell> set_ideal_aggressor [get_nets {n_ideal_aggr1 n_ideal_aggr2}]
1
```

In the following example, the *si_ideal_aggressor* attribute is set using *set_ideal_aggressor* command and then queried using *get_attribute* command. The *get_attribute* command returns *true* for the *si_ideal_aggressor* attribute. The *wire_script* command writes *set_ideal_aggressor* command for net "n_ideal_aggr".

```
pt_shell> get_attribute [get_net n_ideal_aggr] ref_name
n_ideal_aggr

pt_shell> get_attribute [get_net n_ideal_aggr] si_ideal_aggressor
false

pt_shell> set_ideal_aggressor [get_net n_ideal_aggr]
1

pt_shell> get_attribute [get_net n_ideal_aggr] si_ideal_aggressor
true

pt_shell> write_script -format ptsh

 Script contains this command:
 set_ideal_aggressor [get_nets {n_ideal_aggr}]
```

Feedback

See Also

- remove_ideal_aggressor
- get_attribute
- write_script

---

set_ideal_latency

Specifies ideal latency values for the pins in an ideal network.

Syntax

int *set_ideal_latency*

```
[-rise] [-fall]
[-min] [-max]
value
object_list
```

Data Types

value	float
object_list	list

Arguments

`-rise`

Indicates that the *value* option represents the rise latency time. If you do not specify the *-rise* or *-fall* options, both values are set.

`-fall`

Indicates that the *value* option represents the fall latency time. If you do not specify the *-rise* or *-fall* option, both values are set.

`-min`

Indicates that the *value* option represents the minimum latency time. If you do not specify the *-min* or *-max* option, both values are set.

`-max`

Indicates that the *value* option represents the maximum latency time. If you do not specify the *-min* or *-max* option, both values are set.

value

Specifies an ideal latency value on leaf cell pins or top-level ports in an ideal network.

object_list

Specifies a list of leaf cell pins and top-level ports on which ideal latency is set.

Description

Sets an ideal latency value on leaf cell pins and top-level ports of an ideal network.

PrimeTime uses ideal timing for ideal networks, which means that pins and ports have a specified ideal latency (from the *set_ideal_latency* command) or zero ideal latency by default. Ideal networks are normally used during pre-layout to avoid unnecessary DRCs. The specified ideal latency value provides an estimate of the latency time in the ideal network for pre-layout.

The specified latency value overrides the internally-estimated cell and net delay value and any other delay annotations. If the specified pins do not belong to an ideal network, a warning message is generated by the *report_ideal_network* command and the ideal latency *value* option is ignored for those pins. Ideal latency values do not have any effect in ideal clock networks (for example, non-propagated clock networks).

The *set_ideal_latency* command affects all pins in the transitive fanout of the pins or ports on which ideal latency is specified. The total ideal latency at an ideal boundary pin is the sum of latency on the ideal network source and all ideal latencies on the path.

You can use the *set_ideal_latency* command for pins at lower levels of the design hierarchy.

To list ideal latency values, use the *report_ideal_network* command.

To remove the ideal latency values from a design, use the *remove_ideal_latency* or *reset_design* command.

Examples

The following example specifies a rise latency of *1.2* and a fall latency of *0.9* for ports named *A*, *B*, and *C*.

```
pt_shell> set_ideal_latency 1.2 -rise {A B C}
pt_shell> set_ideal_latency 0.9 -fall {A B C}
```

See Also

- remove_ideal_latency
- remove_ideal_network
- remove_ideal_transition
- report_ideal_network
- report_timing

- set_ideal_transition
- set_ideal_network

set_ideal_network

Marks a set of ports or pins in the design as sources of an ideal network. This disables timing update of cells and nets in the transitive fanout of the specified objects.

Syntax

integer *set_ideal_network*

```
[-no_propagate]
object_list
```

Data Types

object_list	list

Arguments

`-no_propagate`

Indicates that the ideal network is not propagated through leaf cells. Ideal properties are enabled on all nets that are electrically connected to the ideal network sources.

object_list

Specifies a list of objects to mark as the sources of an ideal network. Ideal network source objects may be ports or pins of leaf cells at any hierarchical level of the design. If nets are specified in the *object_list*, all of the nets' global driver pins are marked as ideal network sources. The *set_ideal_network* command accepts nets only when you specify the *-no_propagate* option.

Description

This command marks a set of ports or pins in the design as sources of an ideal network. You specify only the source of the network. PrimeTime marks all nets, cells, and pins in the transitive fanout of ideal sources as ideal. The ideal property is automatically spread by the tool and spread again, as necessary, after netlist changes. The criteria for propagating the ideal property, starting at the source pins and ports, are as follows:

- A pin is marked as ideal if you specify it by using the *set_ideal_network* command, if it is either a driver pin and its cell is ideal or if it is a load pin attached to a net that is ideal.
- A net is marked as ideal if all its driving pins are ideal.

- A combinational cell is marked as ideal if at least one of its input pins is ideal and all the other input pins are either ideal, unconnected or attached to a logic constant nets. Objects with the case analysis attribute set are not treated as constant.
- Propagation traverses through combinational cells but stops at sequential cells.
- Design rule checks (DRCs) are disabled on ideal network objects.
- Ideal networks can be specified in the clock or data network.

The latency and transition times of an ideal network are zero by default, but you can override them by using the *set_ideal_latency* and *set_ideal_transition* commands. Ideal timing values do not have any effect in ideal clock networks (for example, non-propagated clock networks).

To reverse the effect of the *set_ideal_network* command, use the *remove_ideal_network* command and specify the source pins or ports for the network. All ideal networks are removed using the *reset_design* command.

Ideal sources, non-source ideal pins with ideal timing values, boundary pins of ideal networks and ideal nets and cells are displayed using the *report_ideal_network* command.

Examples

The following example creates an ideal network on ports in a design called 'TOP'.

```
pt_shell> current_design {TOP}
pt_shell> set_ideal_network {IN1 IN2}
```

The following example creates an ideal network on the multi-driven net named 'netA'.

```
pt_shell> set_ideal_network -no_propagate {netA}
Warning: Transferring ideal net attribute onto driver pin 'U1/Z' of net
 'netA'. (UITE-450)
Warning: Transferring ideal net attribute onto driver pin 'FFD/Q' of net
 'netA'. (UITE-450)
```

See Also

- remove_ideal_network
- report_ideal_network
- reset_design
- set_ideal_latency
- set_ideal_transition

Feedback

set_ideal_transition

Specifies ideal transition values for the pins in an ideal network.

Syntax

int *set_ideal_transition*

```
[-rise] [-fall]
[-min] [-max]
value
object_list
```

Data Types

value	float
object_list	list

Arguments

`-rise`

Indicates that the *value* option represents the rise transition time. If you do not specify the *-rise* or *-fall* option, both values are set.

`-fall`

Indicates that the *value* option represents the fall transition time. If you do not specify the *-rise* or *-fall*option, both values are set.

`-min`

Indicates that the *value* option represents the minimum transition time. If you do not specify the *-min* or *-max*, both values are set.

`-max`

Indicates that the *value* option represents the maximum transition time. If you do not specify the *-min* or *-max* option, both values are set.

value

Specifies an ideal transition value on leaf cell pins or top-level ports in an ideal network.

object_list

Specifies a list of leaf cell pins and top-level ports on which ideal transition is set.

Description

Sets an ideal transition value on leaf cell pins and top-level ports of an ideal network.

Feedback

PrimeTime uses ideal timing for ideal networks, which means that pins and ports have a specified ideal transition (from the *set_ideal_transition* command) or zero ideal transition by default. Ideal networks are normally used during pre-layout to avoid unnecessary DRCs. The specified ideal transition value provides an estimate of the transition time in the ideal network for pre-layout.

The specified transition value overrides the internally-estimated cell and net transition value and any other transition annotations. If the specified pins do not belong to an ideal network, a warning message is generated by the *report_ideal_network* command and the ideal transition *value* option is ignored for those pins. Ideal transition values do not have any effect in ideal clock networks (i.e. non-propagated clock networks).

You can use the *set_ideal_transition* command for pins at lower levels of the design hierarchy.

To list ideal transition values, use the *report_ideal_network* command.

To remove the ideal transition values from a design, use the *remove_ideal_transition* or *reset_design* command.

Examples

The following example specifies a rise transition of *1.2* and a fall transition of *0.9* for ports named *A*, *B*, and *C*.

```
pt_shell> set_ideal_transition 1.2 -rise {A B C}
pt_shell> set_ideal_transition 0.9 -fall {A B C}
```

See Also

- remove_ideal_latency
- remove_ideal_network
- remove_ideal_transition
- report_ideal_network
- report_timing
- set_ideal_latency
- set_ideal_network

---

set_implement_options

Specifies the PrimeECO configuration options for the *check_eco* and *implement_eco* commands.

Feedback

Syntax

```
status set_implement_options
   [-icc2_exec path_name]
   [-icc2_max_cores core_count]
   [-icc2_initialize_script file_name]
   [-icc2_post_link_script file_name]
   [-icc2_pre_legalize_script file_name]
   [-icc2_eco_legalize_script file_name]
   [-icc2_pre_route_script file_name]
   [-icc2_eco_route_script file_name]
   [-icc2_pre_extract_script file_name]
   [-starrc_exec path_name]
   [-starrc_cmd_files file_name_list]
   [-starrc_mode full | incremental]
   [-parasitic_corners pair_list]
   [-parasitic_extractor starrc | icc2]
   [-work_dir directory_name]
   [-insert_fillers]
   [-load_advanced_technology_rules]
```

Data Types

path_name	string
core_count	int
file_name	string
file_name_list	string
pair_list	list
directory_name	string

Arguments

`-icc2_exec` *path_name*

Specifies the path to the icc2_shell executable for IC Compiler II tool.

`-icc2_max_cores` *core_count*

Controls the maximum number of cores that can be used by multicore commands in IC Compiler II.

`-icc2_initialize_script` *file_name*

Specifies the file that contains a custom IC Compiler II script, which is executed during *implement_eco* before opening library.

`-icc2_post_link_script` *file_name*

Specifies the file that contains a custom IC Compiler II script, which is executed during *implement_eco* after the current working block is loaded.

`-icc2_pre_legalize_script` *`file_name`*

Specifies the file that contains a custom IC Compiler II script, which is executed during *implement_eco* before placement legalization.

`-icc2_eco_legalize_script` *`file_name`*

Specifies the file that contains a custom IC Compiler II script, which is executed during *implement_eco* instead of the built-in default placement legalization strategy.

`-icc2_pre_route_script` *`file_name`*

Specifies the file that contains a custom IC Compiler II script, which is executed during *implement_eco* after placement legalization, before ECO routing.

`-icc2_eco_route_script` *`file_name`*

Specifies the file that contains a custom IC Compiler II script, which is executed during *implement_eco* instead of the built-in default ECO routing strategy.

`-icc2_pre_extract_script` *`file_name`*

Specifies the file that contains a custom IC Compiler II script, which is executed during *implement_eco* after ECO routing, before parasitic extraction.

`-starrc_exec` *`path_name`*

Specifies the path to the StarXtract executable for the StarRC tool.

`-starrc_cmd_files` *`file_name_list`*

Specifies the names of one or more StarRC configuration files used to perform parasitic extraction in a signoff environment.

`-starrc_mode full | incremental`

Specifies StarRC parasitic extraction mode. The default is *full*.

`-parasitic_corners` *`pair_list`*

Specifies the parasitic corner name for each active scenario using the following format:

`{{`*`scenario_1 para_corner_1`*`} {`*`scenario_2 para_corner_2`*`} ...}`

`-parasitic_extractor starrc | icc2`

Specifies the Synopsys tool to perform parasitic extraction. The default is *starrc*.

`-work_dir` *`directory_name`*

Specifies the directory where PrimeTime, IC Compiler II, and StarRC exchange data and generate log files. The default is a directory called work_dir created in your current working directory.

Feedback

`-insert_fillers`

Enables stdcell filler insertion during *implement_eco* after ECO routing, before parasitic extraction.

`-load_advanced_technology_rules`

Enables loading advanced technology rules such as spacing rules from the IC Compiler II reference library.

Description

The *set_implement_options* command specifies information to configure PrimeECO operation for physical signoff ECO closure, involving IC Compiler II and StarRC.

You must set these options before you run the *check_eco* command.

For further details, see the *implement_eco* man page.

Examples

The following command sets the path to the IC Compiler II executable and the StarRC executable.

```
prompt> set_implement_options \\
  -icc2_exec /path/to/icc2_shell \\
  -starrc_exec /path/to/StarXtract
```

See Also

- check_eco
- check_routes
- implement_eco
- report_implement_options
- save_block
- set_eco_options
- set_implement_options
- write_implement_changes

---

set_imsa_attributes

Adds, removes, or reports attributes to be saved by the *save_session -only_timing_paths* command and restored by the *restore_session* command into an IMSA session.

Feedback

Syntax

string *set_imsa_attributes*

```
[-add]
[-remove]
[-report]
[-class class_name]
[attribute_list]
```

Data Types

class_name	string
attribute_list	list

Arguments

`-add`

Adds to the list of attributes to be saved and restored into IMSA sessions.

`-remove`

Removes from the list of attributes to be saved and restored into IMSA sessions.

`-report`

Reports the list of additional attributes to be saved and restored into IMSA sessions, as specified previously by the *-add* and *-remove* options, and by usage of the *imsa_save_reporting_attributes* and *imsa_save_eco_data* variables. This is the default action if no options are used in the *set_imsa_attributes* command.

`-class` *class_name*

Specifies the object class of attributes to be added to (or removed from) the list: *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, or *lib_pin*. Specifying the object class is mandatory when you use the *-add* or *-remove* option.

attribute_list

Specifies a list of attributes for the given object class to add to (or remove from) the list of of attributes to be saved and restored into an IMSA sessions.

Description

This command specifes or reports additional attributes to be saved by the *save_session -only_timing_paths* command and restored by the *restore_session* command into an interactive multi-scenario analysis (IMSA) session.

In an IMSA session, timing path collections from multiple scenarios sharing an identical netlist are restored into a single PrimeTime shell session. You can report the timing paths

and query the path attributes in the session, using a limited set of PrimeTime commands. You cannot change constraints or perform timing updates in the session.

By default, the *save_session -only_timing_paths* command saves only the paths in the specified timing path collection, the attributes of those timing paths and their timing points, and the attributes of the associated clocks. No other attributes are saved or restored.

To enable saving and restoring of additional specific object attributes, use the *set_imsa_attributes* with the *-add* option. Specify the object class and the attribute names using the *-class* and *attribute_list* arguments. Use the *-remove* option in a similar manner to remove attributes from the list previously added. You can specify both application attributes and user-defined attributes.

Use the *-report* option to report the current list of attributes to be saved and restored into IMSA sessions. The list is affected by previous usage of the *set_imsa_attributes* command and by the *imsa_save_reporting_attributes* and *imsa_save_eco_data* variable settings.

In the IMSA session, you can use the *get_attribute* command to query the object attributes that were saved and restored into the session.

Examples

The following example shows how to add to the list of saved IMSA attributes and how to report the current list of attributes to be saved:

```
pt_shell> set_imsa_attributes -add -class design max_capacitance
Information: Added attribute max_capacitance of object class design ...
pt_shell> set_imsa_attributes -add -class pin \\
  {max_rise_slack max_fall_slack min_rise_slack min_fall_slack}
Information: Added attribute max_rise_slack of object class pin ...
 ...
pt_shell> set_imsa_attributes -report
 ...
Properties:
    A - Application-defined
    U - User-defined
    I - Importable

Class      Attribute Name           Type          Properties
-----------------------------------------------------------
design     max_capacitance          float         A
pin        max_rise_slack           float         A
pin        max_fall_slack           float         A
pin        min_rise_slack           float         A
pin        min_fall_slack           float         A
```

See Also

- restore_session
- save_session

Feedback

- imsa_save_eco_data
- imsa_save_reporting_attributes
- interactive_multi_scenario_analysis_enabled

---

set_input_delay

Defines the arrival time relative to a clock.

Syntax

string *set_input_delay*

```
[-clock clock_name]
[-reference_pin pin_port_name]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-add_delay]
[-network_latency_included]
[-source_latency_included]
[-sms_scenarios sms_scenarios_list]
delay_value
port_pin_list
```

Data Types

clock_name	list
delay_value	float
pin_port_name	list
port_pin_list	list
sms_scenarios_list	collection

Arguments

`-clock clock_name`

Specifies the clock to which the specified delay is related. If you use the *-clock_fall* option, you must specify the *-clock clock_name* option. If you do not specify the *-clock* option, the delay is relative to time zero for combinational designs. Without using *-clock*, the input delay is not relative to any clock and will not create constrained paths from the port with repspect to any clock. See the variable *timing_input_port_default_clock* to change this behavior.

`-reference_pin pin_port_name`

Specifies the clock pin or port to which the specified delay is related. If you use this option and the propagated clocking is being used, the delay value is

related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The *-network_latency_included* and *-source_latency_included* options cannot be used at the same time as the *-reference_pin* option. For ideal clock network, only source latency is applied.

The pin specified with the *-reference_pin* option should be a leaf pin or port in a clock network, in the direct or transitive fanout of a clock source specified with the *-clock* option. If multiple clocks reach the port or pin where you are setting the input delay and if the *-clock* option is not used, the command considers all of the clocks.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock. When it is used with *-reference_pin* option, the delay is relative to the falling transition of the reference pin. The default is the rising edge or rising transition of a reference pin.

`-level_sensitive`

Specifies that the source of the delay is a level-sensitive latch. This allows the tool to derive setup and hold relationship for paths from this port as if it were a level-sensitive latch. If you do not use the *-level_sensitive* option, the input delay is treated as if it were a path from a flip-flop.

`-rise`

Specifies that *delay_value* refers to a rising transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* option, rising and falling delays are assumed to be equal.

`-fall`

Specifies that *delay_value* refers to a falling transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* option, rising and falling delays are assumed to be equal.

`-max`

Specifies that *delay_value* refers to the longest path. If you do not specify the *-max* or *-min* option, maximum and minimum input delays are assumed to be equal.

`-min`

Specifies that *delay_value* refers to the shortest path. If you do not specify the *-max* or *-min* option, maximum and minimum input delays are assumed to be equal.

`-add_delay`

Specifies whether to add delay information to the existing input delay or to overwrite. Use the *-add_delay* option to capture information about multiple paths leading to an input port that are relative to different clocks or clock edges.

For example, *set_input_delay 5.0 -max -rise -clock phi1 {A}* removes all other maximum rise input delay from "A", because the *-add_delay* option is not specified. Other input delays with different clocks or with clock_fall are removed.

In another example, the *-add_delay* option is specified as *set_input_delay 5.0 -max -rise -clock phi1 -add_delay {A}*. If there is an input maximum rise delay for "A" relative to clock "phi1" rising edge, the larger value is used. The smaller value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is maximum rise input delay relative to a different clock or different edge of the same clock, it remains with the new delay.

`-network_latency_included`

Specifies whether the clock network latency should not be added to the input delay value. If this option is not specified , the clock network latency of the related clock will be added to the input delay value. It has no effect if the clock is propagated or the input delay is not specified with respect to any clock.

`-source_latency_included`

Specifies whether the clock source latency should not be added to the input delay value. If this option is not specified , the clock source latency of the related clock will be added to the input delay value. It has no effect if the input delay is not specified with respect to any clock.

`delay_value`

Specifies the path delay. The *delay_value* option must be in units consistent with the technology library used during analysis. The *delay_value* option represents the amount of time that the signal is available after a clock edge. This usually represents a combinational path delay from the clock pin of a register.

`port_pin_list`

Provides a list of input port or internal pin names in the current design to which *delay_value* is assigned. If you specify more than one object, the objects are enclosed in braces ({}).

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this input delay is applied. When this option is not given the input delay applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Feedback

Description

Sets input path delay values for the current design. Used with the *set_load* and *set_driving_cell* commands, the input and output delays characterize the operating environment of the current design.

The *set_input_delay* command sets input path delays on input ports relative to a clock edge. Unless specified, input ports are assumed to have zero input delay. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay from a level-sensitive latch, use the *-level_sensitive* option. If the latch is positive-enabled, set the input delay relative to the rising clock edge. If the latch is negative enabled, set the input delay relative to the falling clock edge. If time is borrowed at that latch, add that time borrowed to the path delay from the latch when determining input delay.

The *characterize_context* command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

If for a specific clock the maximum delay is less than minimum delay, PrimeTime makes the maximum delay equal to that of the minimum delay.

PrimeTime adds input delay to path delay for paths starting at primary inputs, and to output delay for paths ending at primary outputs.

PrimeTime allows an input port to behave simultaneously as a clock and data port. You can use the *timing_simultaneous_clock_data_port_compatibility* variable to enable or disable the simultaneous behavior of the input port as a clock and data port. When this variable is set to its default of *false*, simultaneous behavior is enabled and you can use the *set_input_delay* command to define the timing requirements for input ports relative to a clock. In this situation, the following applies:

- If you specify the *set_input_delay* command relative to a clock defined at the same port and the port has data sinks, the command is ignored and an error message is issued. There is only one signal coming to port, and it cannot be at the same time data relative to a clock and the clock signal itself.
- If you specify the *set_input_delay* command relative to a clock defined at a different port and the port has data sinks, the input delay is set and controls data edges launched from the port relative to the clock.
- Regardless of the location of the data port, if the clock port does not fanout to data sinks, the input delay on the clock port is ignored and you receive an error message.

When you set the *timing_simultaneous_clock_data_port_compatibility* variable to *true*, the simultaneous behavior is disabled and the *set_input_delays* command defines the arrival time relative to a clock. In this situation, when an input port has a clock defined on it, PrimeTime considers the port exclusively as a clock port and imposes restriction on the

data edges that are launched. PrimeTime also prevents setting input delays relative to another clock.

To control the clock source latency for any clocks defined on an input port, you must use the *set_clock_latency* command.

If a reference pin is not reachable by any given clock, zero arrival time is assumed. If no clock is specified with a reference pin and this reference pin is not reachable by any clock, an unconstrained path is reported. This behavior is changed after version X-2005.06. The new behavior is these invalid constraints are ignored and the path is constrained by whatever it should be as if no such constraints are defined at all. The *check_timing* command generates a warning if the reference pin is not reachable by any active clock, or if multiple clocks reach the reference pin and no clock is specified in the command.

To get a report on the latency calculation using a reference pin, use one of the following commands:

```
report_timing -path_type full_clock
report_timing -path_type full_clock_expanded
```

To list input delays associated with ports, use the *report_port* command. To list input delays of internal pins, use the *report_design* command. To remove input delay values, use the *remove_input_delay* or *reset_design* command.

Examples

The following example sets an input delay of 2.3 for ports "IN1" and "IN2" on a combinational design. Because the design is combinational, no clock is needed.

```
pt_shell> set_input_delay 2.3 {IN1 IN2}
```

The following example sets input delay of 1.2 relative to the rising edge of "CLK1" for all input ports in the design.

```
pt_shell> set_input_delay 1.2 -clock CLK1 [all_inputs]
```

The following example sets the input and output delays for the bidirectional port "INOUT1". The input signal arrives at "INOUT1" 2.5 units after the falling edge of "CLK1". The output signal is required at "INOUT1" at 1.4 units before the rising edge of "CLK2".

```
pt_shell> set_input_delay 2.5 -clock CLK1 -clock_fall {INOUT1}

pt_shell> set_output_delay 1.4 -clock CLK2 {INOUT1}
```

The following example models the situation where there are three paths to input port "IN1". The first path is relative to the rising edge of "CLK1". The second path is relative to the falling edge of "CLK1". The third path is relative to the falling edge of "CLK2". The *-add_delay* option is used to indicate that new input delay information does not cause old information to be removed.

```
pt_shell> set_input_delay 2.2 -max clock CLK1 -add_delay {IN1}

pt_shell> set_input_delay 1.7 -max clock CLK1 -clock_fall -add_delay
 {IN1}

pt_shell> set_input_delay 4.3 -max clock CLK2 -clock_fall -add_delay
 {IN1}
```

In the following example, two different maximum delays and two minimum delays for port "A" are specified using the *-add_delay* option. Since the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained (in this case, 5.0 and 1.1). If the *-add_delay* option is not used, the new information overwrites the old information.

```
pt_shell> set_input_delay 3.4 -max -clock CLK1 -add_delay {A}

pt_shell> set_input_delay 5.0 -max -clock CLK1 -add_delay {A}

pt_shell> set_input_delay 1.1 -min -clock CLK1 -add_delay {A}

pt_shell> set_input_delay 1.3 -min -clock CLK1 -add_delay {A}
```

See Also

- all_inputs
- characterize_context
- check_timing
- create_clock
- current_design
- remove_input_delay
- report_design
- report_port
- report_timing
- reset_design
- set_driving_cell
- set_load
- set_output_delay

Feedback

set_input_noise

Sets a noise bump for a pin or port.

Syntax

int *set_input_noise*

```
[-add_noise]
[-above]
[-below]
[-low]
[-high]
[-height width_value]
[-width  height_value]
object_list
```

Data Types

width_value	float
height_value	float
object_list	list

Arguments

`-add_noise`

Specifies whether to add noise information to the existing input noise or to overwrite it. For example, if the *set_input_noise* is used previously, another *set_input_noise* with the *-add_noise* option will sum the noise information with value previously set by the *set_input_noise* command.

`-above`

Specifies a noise bump for above ground or power rail noise analysis region.

`-below`

Specifies a noise bump for below ground or power rail noise analysis region.

`-low`

Specifies a noise bump for ground rail noise.

`-high`

Specifies a noise bump for power rail noise.

`-height height_value`

Specifies the height of the noise bump. The height is in the voltage units of the library.

`-width` *`width_value`*

Specifies the width of the noise bump. The width is in the time units of the library. 1.0.

`object_list`

Specifies a list of pins or ports.

Description

Rather than allow PrimeTime SI to calculate propagated noise bumps, you can explicitly specify a noise bump at any port or pin. In this command, you specify the port or pin in the design on which to apply the noise and the characteristics of the noise bump: the type (above/below high/low), the width in library time units, the height in library voltage units (always entered as a positive number).

PrimeTime SI treats this injected noise as propagated noise from the driver of the net connected to the pin. This noise bump overrides any propagated noise that would otherwise be calculated from the driver of the net.

You can specify propagated noise on an output pin rather than a load pin. In that case, the specified noise bump overrides any propagated noise that would otherwise be calculated from the driver. The specified noise bump is propagated through the subnets and attenuated by the parasitic capacitance and resistance specified for the net, until the propagated noise reaches each of load pin on the net.

The *report_noise_calculation* command shows whether input noise information was calculated or annotated by this command.

Examples

This example specifies a noise bump height of 0.58, width of 1.70 for pin B of cell U1 in the current design.

```
pt_shell> set_input_noise -above -low -width 1.70 -height 0.58\\
 [get_pins U1/B]
```

See Also

- report_noise_calculation
- remove_input_noise

---

set_input_transition

Sets a fixed transition time on input or inout ports.

Feedback

Syntax

string *set_input_transition*

```
[-rise]
[-fall]
[-min]
[-max]
[-clock clock_name]
[-clock_fall]
[-sms_scenarios sms_scenarios_list]
transition
port_list
```

Data Types

clock_name	string
transition	float
port_list	list

Arguments

`-rise`

Sets only the rise transition.

`-fall`

Sets only the fall transition.

`-min`

Sets transition for minimum conditions.

`-max`

Sets transition for maximum conditions.

`-clock` *clock_name*

The input transition is set relative the specified clock. This option is deprecated, and will be ignored.

`-clock_fall`

Specifies that the transition is relative to the falling edge of the clock. The default is the rising edge. This option is deprecated, and will be ignored.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios for which the specification applies. When this option is not given, the specification applies to all SMS scenarios. This collection is created by the *get_sms_scenarios* command.

Only scenarios relating to the corner domain and voltage domain at the port are honored. Scenario specifications for other corner or voltage domains are accepted by the command but ignored when propagating transitions.

transition

Port transition value. This is a floating point number greater than or equal to zero.

port_list

A list of input or inout ports.

Description

The *set_input_transition* command specifies a fixed transition time for a list of input or inout ports. This transition time is not affected by the capacitance of the net connected to the port. The transition time calculates delays for nets and cells in the transitive fanout of the port. The port itself has no cell delay.

To display port transition or drive capability information, use the *report_port -drive* command.

There are two other methods of describing port drive capability. The *set_driving_cell* command causes the port to have transition time calculated as if a given library cell was driving the net. The *set_driving_cell* command also has a cell delay equal to the load-dependent portion of the delay driving the net of the library cell. The driving cell approach is accurate for nonlinear delay models even if the capacitance is changed. Another method is to use the *set_drive* command, which models the driver as a linear resistance. The most recent drive command has precedence.

Examples

This example specifies that ports matching the pattern "DATA_IN*" has a transition time of 0.75 units.

```
pt_shell> set_input_transition 0.75 [get_ports DATA_IN*]
```

See Also

- set_driving_cell
- set_drive
- report_port
- set_clock_transition

Feedback

set_isolation

Defines the UPF isolation strategy for the power domains in the design.

Syntax

int *set_isolation*

```
-domain power_domain
[-isolation_power_net supply_net_name]
[-isolation_ground_net supply_net_name]
[-isolation_supply_set supply_set_name]
[-no_isolation]
[-elements objects]
[-clamp_value 0 | 1 | latch | Z]
[-applies_to input | output | both]
[-applies_to_boundary upper | lower | both]
[-source source_supply_set_name]
[-sink sink_supply_set_name]
[-diff_supply_only TRUE | FALSE]
[-isolation_signal signal]
[-location self | parent | fanout]
[-isolation_sense sense]
[-force_isolation]
[-name_prefix prefix_string]
[-name_suffix suffix_string]
isolation_strategy
```

Data Types

```
power_domain                string
supply_net_name             string
supply_set_name             string
objects                     list
source_supply_set_name      string
sink_supply_set_name        string
prefix_string               string
suffix_string               string
isolation_strategy          string
signal                   string
sense                    string
location                    string
```

Arguments

`-domain power_domain`

Specifies the power domain name that this UPF isolation strategy is applied to.

`-isolation_power_net supply_net_name`

Specifies the isolation power net for the isolation cells that are created based on this UPF isolation strategy.

`-isolation_ground_net` *`supply_net_name`*

Specifies the isolation ground net for the isolation cells that are created based on this UPF isolation strategy. At least one of the *-isolation_power_net* or *-isolation_ground_net* options should be specified if the *-no_isolation* option is not specified.

`-isolation_supply_set` *`supply_set_name`*

Specifies the supply set whose power and ground functions are to be used as the isolation power and ground nets respectively. This option is mutually exclusive with the *-isolation_power_net* and *-isolation_ground_net* options.

`-no_isolation`

Specifies that elements under the influence of this *set_isolation* command should not be isolated.

`-elements` *`objects`*

Specifies the objects that this UPF isolation strategy are applied to. The objects can be pins of the root cells of the power domain, ports of the top-level design for top-level power domains.

`-clamp_value 0 | 1 | latch | Z`

Specifies the clamp value of the isolation cells that should be created based on this strategy. The default of this option is 0.

`-applies_to input | output | both`

Specifies whether the given *set_isolation* command applies to all inputs or outputs or both kind of ports of the power domain. The default of this option is *output*.

`-applies_to_boundary upper | lower | both`

Specifies explicitly whether to apply the strategy to upper boundaries(lowConns) only, to lower boundaries(highConns) only, or to both upper and lower boundaries of the power domain. PT/PrimePower does not use this option for any analysis. PrimeTime ignores this option.

`-source` *`source_supply_set_name`*

Specifies the source supply set that applies on the elements of the strategy. It filters the ports/pins receiving a net that is driven by logic powered by the supply set. This option is mutually exclusive with the *-applies_to* option.

`-sink` *`sink_supply_set_name`*

Specifies the sink supply set that applies on the elements of the strategy. It filters the ports/pins driving a net that fans out to logic powered by the supply set. This option is mutually exclusive with the *-applies_to* option.

`-diff_supply_only TRUE | FALSE`

Determines the isolation behavior between driver and receiver supply sets. If the driver is powered by the same supply set as a receiver of the port, no isolation cell is introduced into the path. The default for this option is *FALSE*. The *-source*, *-sink*, and *-diff_supply_only* options are mutually exclusive.

`-isolation_signal` *`signal`*

Specifies the isolation signal to use as the control signal of the isolation cells that should be created as a result of this isolation strategy. The *signal* value can be a pin, port, or net.

`-location self | parent | fanout`

Specifies the hierarchical location of the isolation cells that should be created as a result of this isolation strategy. The default value is *self*.

A value of *self* specifies that the isolation cell is placed in the hierarchy of the port being isolated.

A value of *parent* specifies that isolation cells is added in the parent hierarchy of the isolating port's hierarchy.

A value of *fanout* specifies that isolation cells is added in the fanout of the isolating port.

`-isolation_sense` *`sense`*

Specifies the isolation sense of the isolation cells that should be created as a result of this isolation strategy. Excepted values are either *low* or *high*. The default value is *high*.

`-force_isolation`

Forces isolation.

`-name_prefix` *`prefix_string`*

Prepends the specified prefix to the names of generated isolation instances or nets related to implementation of the isolation strategy. This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-name_suffix` *`suffix_string`*

Appends the specified suffix to the names of generated isolation instances or nets related to implementation of the isolation strategy. This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

isolation_strategy

Specifies the name of the UPF isolation strategy. The name must be unique within the specified power domain.

update

Specifies the elements to be updated for existing isolation.

Description

This command defines the UPF isolation strategy for the ports of the specified power domain.

If *-elements* and *-applies_to* options are not specified, the isolation strategy is applied to all the outputs ports of the power domain.

If neither the *-isolation_supply_set*, *-isolation_power_net*, nor *-isolation_ground_net* option is specified, the isolation power and ground nets are automatically set to the power and ground functions of the *default_isolation* handle of the power domain, for which the strategy is being defined.

PrimeTime uses the *set_isolation* command to build virtual power ground connectivity. The command creates explicit connection for isolation power and ground nets to power and ground pins of isolation cells. If a cell matches multiple isolation rules (*set_isolation* and *set_isolation_control* commands), the last entered *set_isolation* command rule is used to derive the PG connectivity of the cell. Use the *upf_isolation_strategy* cell attribute to check which isolation rule was applied to the cell.

Examples

The following example shows how to define a UPF isolation strategy for all the output ports of the specified power domain PD1. Power domain PD1 is defined on the shutdown_inst instance.

```
pt_shell> set_isolation isolation_1 -domain PD1 \\
            -isolation_power_net PN1 \\
            -isolation_ground_net GN1
```

The following example shows how to define a UPF isolation strategy for specific ports of the specified power domain.

```
pt_shell> set_isolation isolation_2 -domain PD1 \\
            -isolation_power_net PN1 \\
            -isolation_ground_net GN1 \\
            -elements shutdowm_inst/special_port
```

The following example shows how to find all isolation cells in the design and the strategy applied to each cell:

```
pt_shell> foreach_in_collection c [sort_collection [get_cells
 -hierarchical *
            -filter "upf_isolation_strategy != {}"] full_name] {
               echo [format {  Cell  %-8s  isolation %s}
                  [get_attribute $c full_name] [get_attribute $c
 upf_isolation_strategy]]
            }
```

The following example shows how to define a UPF isolation strategy using a supply set:

```
pt_shell> set_isolation isolation_2 -domain PD1 \\
            -isolation_supply_set iso_supply_set \\
            -elements shutdown_inst/special_port
```

The following example shows how to define a UPF isolation strategy without using a supply set or supply nets.

```
pt_shell> set_isolation isolation_2 -domain PD1 \\
            -elements shutdown_inst/special_port
```

The following example shows how to define a UPF isolation strategy using the *-source* and *-sink* options.

```
pt_shell> set_isolation isolation_2 -domain PD1 \\
            -isolation_supply_set iso_supply_set \\
            -source PD_primary_set \\
            -sink sset
```

The following example shows how to define a UPF isolation strategy using the *-diff_supply_only* option.

```
pt_shell> set_isolation isolation_2 -domain PD1 \\
            -isolation_supply_set iso_supply_set \\
            -source PD_primary_set \\
            -diff_supply_only TRUE
```

The following example shows how to update elements using the *-update* option.

```
pt_shell> set_isolation isolation_2 -domain PD1 \\
            -element PD1_INST/INPUT \\
            -isolation_supply_set  SS1 \\
            -update PD2_INST/INPUT
```

See Also

- create_power_domain
- create_supply_net

Feedback

set_isolation_control

Provides additional options needed for creating isolation cells. This command is needed with most *set_isolation* commands.

Syntax

status *set_isolation_control*

```
-domain domain
-isolation_signal signal
[-location self | parent | fanout]
[-isolation_sense sense]
isolation_name
```

Data Types

```
domain                  string
signal                  string
sense                   string
isolation_name          string
```

Arguments

`-domain domain`

Specifies the power domain name to which this UPF isolation strategy is applied.

This option is required.

`-isolation_signal signal`

Specifies the isolation signal to use as the control signal of the isolation cells that should be created as a result of this isolation strategy. The *signal* value can be a pin, port, or net.

This option is required.

`-location self | parent | fanout`

Specifies the hierarchical location of the isolation cells that should be created as a result of this isolation strategy. The default value is *self*.

A value of *self* specifies that the isolation cell is placed in the hierarchy of the port being isolated.

A value of *parent* specifies that isolation cells is added in the parent hierarchy of the isolating port's hierarchy.

A value of *fanout* specifies that isolation cells is added in the fanout of the isolating port.

`-isolation_sense` *sense*

Specifies the isolation sense of the isolation cells that should be created as a result of this isolation strategy. Excepted values are either *low* or *high*. The default value is *high*.

isolation_name

Specifies the UPF isolation strategy name. The isolation strategy should already be defined in the specified power domain.

This argument is required.

Description

This command applies to an existing isolation strategy specified by the *set_isolation* command. It provides extra parameters needed for creating isolation cells.

PrimeTime uses additional information about the isolation strategy provided by this command to properly match isolation strategies to individual leaf cells (PrimeTime does not create isolation cells). Isolation strategy is used for deriving PG connectivity from UPF description.

Examples

The following example shows how to define a UPF isolation strategy for all output ports of the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst, and isolation strategy isolation_1 is already defined

```
pt_shell> set_isolation_control isolation_1 -domain PD1 \\
            -location parent \\
            -isolation_signal en \\
            -isolation_sense high \\
```

See Also

- set_isolation

---

set_latch_loop_breaker

Specifies transparent latch data pins to be used as loop breaker latch data pins when the *timing_enable_through_paths* variable is set to *true*.

Feedback

Syntax

status *set_latch_loop_breaker*

```
-pin pin_list
[-remove]
[-avoid]
```

Data Types

pin_list	list

Arguments

`-pin` *pin_list*

Specifies data pins of transparent latches to be loop breakers.

`-remove`

Removes the user-specified setting on pins.

`-avoid`

Avoids using the specified pins as loop breakers. Requests to avoid using specified pins as loop breaker pins cannot always be honored, especially if there is a loop from the pin through combinational logic to itself.

Description

When sequential loops of transparent latches exist in a design, the tool selects certain latch data pins for special analysis as loop breaker data pins. Reporting paths through these latch data pins is not permitted except by using the *-trace_latch_borrow* option of the *report_timing* command.

Use the *set_latch_loop_breaker* command to guide the tool in selecting data pins as loop breaker data pins. By guiding the tool to select some latch data pins as loop breakers and not others, you can report the paths of interest.

Before using the *set_latch_loop_breaker* command, set the *timing_enable_through_paths* variable to *true*.

Examples

The following example specifies that the L1/D pin is a loop breaker pin.

```
pt_shell> set_latch_loop_breaker -pin [get_pin L1/D]
```

Feedback

See Also

- get_latch_loop_groups
- report_latch_loop_groups
- timing_enable_through_paths

---

set_level_shifter_strategy

Sets the type of strategy to use for reporting the signal level mismatches in the design.

Syntax

status *set_level_shifter_strategy*

`[-rule strategy]`

Data Types

`strategy` `string`

Arguments

`-rule strategy`

Specifies types of source-sink pairs of mismatching signal levels that the *check_timing -include signal_level* command reports. The following values are allowed:

- *all* (the default) - Reports all voltage level mismatches.
- *low_to_high* - Reports the voltage level mismatches when a source at a lower voltage drives a sink at a higher voltage.
- *high_to_low* - Reports the voltage level mismatches when a source at a higher voltage drives a sink at a lower voltage.

Description

This command specifies the strategy for reporting nets that need insertion of level-shifters.

Examples

The following example sets the level-shifter strategy to the *high_to_low* option:

```
prompt> set_level_shifter_strategy -rule high_to_low
```

Feedback

See Also

- check_timing
- set_level_shifter_threshold

---

set_level_shifter_threshold

Sets the minimum threshold beyond which the voltage adjustment is required.

Syntax

integer *set_level_shifter_threshold*

```
-voltage volt
-percent diff
```

Data Types

```
volt      float
diff      float
```

Arguments

`-voltage volt`

The absolute difference between the source and sink voltages.

`-percent diff`

The percentage by which the source and sink voltages must differ. The percentage is determined as follows:

```
abs(driver(v)-load(v))/driver(v)*100
```

Description

This command specifies the minimum threshold value for the voltage difference between a source and a sink. Voltage differences above the minimum threshold are reported by *check_timing -include signal_level*. The threshold can be specified as the absolute difference in voltages or a percentage difference or both.

When both *-voltage* and the *-percent* options are both specified: Since physical tools insert level shifters when either absolute or relative threshold is exceeded the *check_timing -include signal_level* only skips signal level mismatches that are smaller than both the absolute and the relative thresholds.

The default threshold voltage and percentage is zero. Therfore you typically must specify both absolute and relative thresholds to suppress reporting of below-threshold level mismatches.

Feedback

Examples

The following example sets the level shifter threshold value to the absolute difference of 0.1 (and sets large relative threshold):

```
psyn_shell-t> set_level_shifter_threshold -voltage 0.1 -percent 100
```

The following example sets the threshold value 5 percent (and sets large absolute threshold):

```
prompt> set_level_shifter_threshold -percent 5 -voltage 10
```

The following example sets the threshold value to 0.1 difference and 5 percent, so if either value is reached, the signal level mismatch is reported:

```
prompt> set_level_shifter_threshold -voltage 0.1 -percent 5
```

See Also

- check_timing
- set_level_shifter_strategy

---

set_lib_cell_spacing_label

Sets an inter cell spacing constraint label on a reference cell in preparation for defining minimum spacing rules between library cells.

Syntax

status *set_lib_cell_spacing_label*

```
-name label_name
[-left_lib_cells list_of_lib_cells]
[-right_lib_cells list_of_lib_cells]
```

Data Types

```
label_name           string
list_of_lib_cells    list
```

Arguments

`-name label_name`

Inter cell constraint label to assign to one or more reference cells.

`-left_lib_cells list_of_lib_cells`

Specifies a list of reference cells that have the given label assigned to the left side of the cell in the "North" orientation. The argument must be a string list of library cells.

`-right_lib_cells` *`list_of_lib_cells`*

Specifies a list of reference cells that have the given label assigned to the right side of the cell in the "North" orientation. The argument must be a string list of library cells.

Description

This command assigns a label to a list of reference (library) cells to set inter cell spacing constraints to be enforced by the legalization tool. A label is a string that you assign to the left or right side of a standard cell.

This command works together with the *set_spacing_label_rule* command, which defines the illegal range of spacing between two adjacent labels. All spacing rules specified by *set_spacing_label_rule* command must be satisfied to produce a legal placement.

With inter cell spacing rules specified, standard cells can be placed next to each other only when they do not violate the spacing rules.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of standard cells. The labels and spacing rules are stored in the library and are applied to all designs using the library.

Examples

The following command assigns the label "X" to reference cells AND* on the left side of the cell, and to reference cells NAND* on the right side of the cell (for cells in the "North" orientation).

```
prompt> set_lib_cell_spacing_label -name {X} \ -left_lib_cells {AND*} \ -right_lib_cells {NAND*}
```

The following command assigns the labels "Y" and "Z" to the right side of reference cell OR2X4.

```
prompt> set_lib_cell_spacing_label -name {Y Z} -right_lib_cells {OR2X4}
```

See Also

- set_spacing_label_rule

---

set_lib_rail_connection

Sets a physical power pin on a library cell.

Syntax

status *set_lib_rail_connection*

Feedback

```
-lib_cell_names lib_cells
-lib_pin_names library_pin_names
-lib_rail_names power_rail_names
```

Data Types

```
lib_cells              list
library_pin_names      list
power_rail_names       list
```

Arguments

`-lib_cell_names lib_cells`

Recognizes the specified list of library cells on power rails.

`-lib_pin_names library_pin_names`

Annotates the specified library pin names.

`-lib_rail_names power_rail_names`

This argument is accepted but not currently supported in PrimeTime.

Description

The *set_lib_rail_connection* command associates the name of a power pin with a library cell. When linking physical Verilog to a logic library, pins specified by this command can be present in the Verilog but not in the logic library.

Examples

The following example shows how to set a physical power pin on a library cell to link with to logic library.

```
pt_shell> set_lib_rail_connection -lib_cell_names typ/INVX1
 -lib_pin_names VDD1
```

When subsequently linking physical Verilog, pins named VDD1 on the INVX1 library cell are ignored when linking to the typ logic library. This allows successful linking of physical Verilog to logic libraries in which power pins not specified.

See Also

- read_verilog

---

set_library_driver_waveform

Sets the driver waveform used to characterize the timing library.

Feedback

Syntax

status *set_library_driver_waveform*

```
-type ramp | standard
  [lib_objs]
```

Data Types

```
lib_objs          list
```

Arguments

```
-type ramp | standard
```

Specifies the type of the waveforms:

- *ramp* - Linear waveforms.
- *standard* - Waveforms commonly known as Synopsys predriver.

lib_objs

List of libraries or library cells on which the driver waveform applies.

Description

The data tables in the timing libraries are indexed by slew. The library characterization process could use the standard Synopsys predriver waveform or a simple ramp waveform to characterize the library. The library data reflects the type of waveform used for characterization. The shape of the waveform generally has only a small effect on the timing results. However, for some forms of advanced analysis, the waveform shape can become significant:

- For advanced crosstalk delay analysis using a CCS noise library, using the exact waveform shape in PrimeTime SI is necessary to get the best possible accuracy.
- When the *write_spice_deck* command writes out a SPICE deck for a given path or arc, and advanced waveform propagation is not enabled, each input to the cell should use the same waveform that was used to characterize the library, to get the best possible correlation between PrimeTime SI and SPICE simulation results.

This command specifies the waveform shape for the library objects. This command allows two type of waveforms: *ramp* and *standard* (Synopsys predriver waveform). If the library object is not provided the waveform is applied to the whole design. When waveform specified on a library cell it gets higher priority than the waveform set on library and the waveform set on library gets higher priority than the design. This command triggers the *update_timing* command.

Feedback

Examples

This example specifies how to set ramp for the whole design. When the *write_spice_deck* command generates the SPICE deck, it uses ramp for voltage sources if no other waveform is specified on the library or library cell.

```
pt_shell> set_library_driver_waveform  -type ramp
```

In the following example the *libOld* is characterized with ramp and the *libNew* library is characterized with standard waveforms. It shows how to set these waveforms after the libraries read to PrimeTime. The *lib_pin* attribute of the *driver_waveform_fall/ driver_waveform_rise* could be used to get the name of the characterization waveforms.

```
pt_shell> set_library_driver_waveform -type ramp [get_lib libOld]
pt_shell> set_library_driver_waveform -type standard [get_lib libNew]
pt_shell> get_attr [get_lib_pin libOld/INV/A] driver_waveform_fall
pt_shell> get_attr [get_lib_pin libOld/INV/A] driver_waveform_rise
```

See Also

- update_timing
- write_spice_deck
- delay_calc_waveform_analysis_mode

---

set_license_limit

Specifies the upper limit on the number of licenses of given features that can be checked out.

Syntax

```
status set_license_limit
  feature_list
  -quantity num_licenses
```

Data Types

feature_list	list
num_licenses	integer

Arguments

feature_list

Specifies a list of features to which the limit applies.

-quantity *num_licenses*

Specifies the maximum number of licenses that can be checked out.

Feedback

Description

The *set_license_limit* command specifies the upper limit on the number of licenses of given features that can be checked out.

Licenses can be checked out explicitly using the *get_license* command or implicitly by the tool itself.

If the given license limit is lower than the number of licenses already checked out, the command will try to reduce the number of checked out licenses to the limit.

To remove the license limit, use the *remove_license_limit* command. Currently defined license limits can be reported using the *report_license_limit* command.

Examples

In the following example, the upper limit on the number of PrimeTime licenses is set to seven and the number of PrimeTime SI licenses set to four. No licenses are checked out at the point at which the commands are called, the licenses are checked out as and when they are needed during the subsequent analysis.

```
pt_shell> set_license_limit PrimeTime -quantity 7
Information: Setting license limit for feature 'PrimeTime' to 7. (PT-021)
1
pt_shell> set_license_limit PrimeTime-SI -quantity 4
Information: Setting license limit for feature 'PrimeTime-SI' to 4.
 (PT-021)
1
```

See Also

- get_license
- remove_license
- remove_license_limit
- report_license_limit
- list_licenses
- report_multi_scenario_design

---

set_link_lib_map

Set the library map for a design or for an instance or for a referece using a file.

Syntax

string *set_link_lib_map*

```
link_lib_map_file
[-reference module_name]
[-instance instance_name]
```

Data Types

```
link_lib_map_file     string
module_name           string
instance_name         string
```

Arguments

`link_lib_map_file`

Specifis the mapping file which contains the library mapping for modules or instances.

`-reference`

Name of the verilog module to which the mapping file is to be applied.

`-instance`

Name of the cell instance to which the mapping file is to be applied.

Description

The set_link_lib_map command is used to specify the library mapping to the instances or verilog modules. This is used as an alternative to the link_path_per_instance setting. Hence the setting and the set_link_lib_map commands are mutually exclusive. An LNK-068 is thrown if link_path_per_instance is already set.

The link_lib_map_file should contain the library mapping in the format: [instace_name : [list_of_libraries]. An entry, '*' is allowed in list of libraries, which means to pick up from the memory.

By default link_lib_map_file is used for the mappping to the top_design, if none of -reference or -instance options are specifed. The options "-reference" and "-instance" are mutually exclusive. When one of them is specified link_lib_map_file is considered for the specified module or instance.

Examples

The following scripts illustrates the set_link_lib_map command.

```
set link_path "* my_linked_lib.db"
set_link_lib_map -ref test $map_path/test_map
read_verilog ./test.v
current_design test
link

set link_path "* my_linked_lib.db"
set_link_lib_map -instance V2 $map_path/test_map
```

```
read_verilog ./test.v
current_design test
link
```

See Also

- current_design
- read_verilog
- link_path

---

set_load

Sets the capacitance on the specified ports and nets in the current design.

Syntax

```
status set_load
  [-min]
  [-max]
  [-rise]
  [-fall]
  [-pin_load]
  [-wire_load]
  [-subtract_pin_load]
  [-allow_negative_cap]
  capacitance
  objects
```

Data Types

```
capacitance   float
objects       list
```

Arguments

`-min`

Indicates that the *capacitance* is the minimum capacitance. Applies only to designs in min-max mode (minimum and maximum operating conditions).

`-max`

Indicates that the *capacitance* is the maximum capacitance. Applies only to designs in min-max mode (minimum and maximum operating conditions).

`-rise`

Indicates that the *capacitance* is the rise capacitance. This option can be used in combination with the *-min* or *-max* option. Use this option only with ports. An error message is generated if the *objects* list contains nets.

`-fall`

Indicates that the *capacitance* is the fall capacitance. This option can be used in combination with the *-min* or *-max* option. Use this option only with ports. An error message is generated if the *objects* list contains nets.

`-pin_load`

Indicates that the specified *capacitance* is a pin capacitance. Pin capacitance is not subject to "scaling" using tree_type. Use this option only with ports. An error message is generated if the *objects* list contains nets.

If you do not specify either the *-pin_load* or *-wire_load* option or specify both options, the *-pin_load* option is the default.

`-wire_load`

Indicates that the specified *capacitance* is a wire capacitance. Pin capacitance is subject to "scaling" using tree_type. Use this option only with ports. An error message is generated if the *objects* list contains nets.

If you do not specify either the *-pin_load* or *-wire_load* option or specify both options, the *-pin_load* option is the default.

`-subtract_pin_load`

Indicates that the current pin capacitances of the specified net are to be subtracted from *capacitance* before the net capacitance value is set. Any resulting negative net capacitance values are set to zero. Use this option only if *capacitance* includes pin capacitances.

`-allow_negative_cap`

Allows the command allowed to set a negative capacitance. Use this option only with the *-subtract_pin_load* option. After subtracting the pin capacitance, the resulting capacitance can become negative.

`capacitance`

Specifies the capacitance value to be set on the ports and nets in *objects*. It is in the units of the logic library. The capacitance value must be greater than or equal to zero. Any negative value is set to zero.

`objects`

Specifies a list of ports and nets in the current design, on which the *capacitance* is to be set. If you specify a name (for example, *net_name*) instead of a real object_list (for example, by using the *get_port* command with the *port_name* option or the *get_net* command with the *net_name* option), the tool first searches the list of all ports and then searches the list of all nets for a match.

`-sms_scenarios`

Limits the specification to only the provided SMS scenarios. When this option is not given, the specification applies to all SMS scenarios. Use the *get_sms_scenarios* command to obtain SMS scenarios.

Only *-pin_load* values support this option. For *-wire_load* values, the last specified value applies to all SMS scenarios.

Description

This command sets the capacitance to a specified value on specified ports and nets in the current design. If the current design is hierarchical, you must link it using the *link_design* command.

Depending on the options used, the *set_load* command sets these attributes on the specified objects:

```
wire_capacitance_max
wire_capacitance_min
pin_capacitance_max
pin_capacitance_min
```

If this command is run without any options, the *pin_capacitance_max* attribute is set; in min-max mode, the *pin_capacitance_min* attribute is also set.

For ports, if the *set_load* command is issued with only the *-wire_load* option, the *wire_capacitance_max* attribute is set on the specified ports; in min-max mode, the *wire_capacitance_min* attribute is also set. However, the value is actually counted as part of the total wire capacitance and not as part of the pin/port capacitance.

In min-max mode, using either the *-min* or *-max* option sets only the **_min* or **_max* attributes, respectively.

If the *-rise* option is specified, the *pin_capacitance_max_rise* and *pin_capacitance_min_rise* attributes are set. You can use this option in conjunction with either the *-min* or *-max* option to set only one of the two attributes. The same conditions apply for the *-fall* option and the *pin_capacitance_*_fall* attributes.

The total capacitance on a net is the sum of all pin capacitances, port capacitances, and wire capacitances associated with that net. The specified *capacitance* overrides both the internally-estimated net capacitance value and any net capacitance set by the *read_parasitics* command.

You can also use the *set_load* command for nets at lower levels of the design hierarchy. These nets are specified in the "BLOCK1/BLOCK2/NET_NAME" format. The tool treats capacitances in such a way that timing on a subblock should be the same as timing on the higher-level design, since pin/wire caps on ports represent a part of the higher-level design.

To view capacitance values on ports and nets, use the *report_port* and *report_net* commands, respectively. To remove a capacitance value, use the *remove_capacitance* command; you can remove annotated capacitances from the full design by using the *reset_design* command.

Examples

The following example sets a capacitance of 2 units on the xyz port.

```
pt_shell> set_load 2 xyz
```

The following example sets a capacitance of 2.5 units (the estimated wire capacitance) plus the capacitance of three inverter input pins from the tech_lib library named on all output ports. It uses the user-defined *port_capacitance* variable to store this capacitance value.

```
pt_shell> set pin_cap \\
            [get_attribute [get_lib_pins tech_lib/IV/A] pin_capacitance]
2.0
pt_shell> set port_capacitance [expr 2.5 + (3 * $pin_cap)]
8.5
pt_shell> set_load $port_capacitance [all_outputs]
1
```

The following example uses the *get_attribute* command to get the value of the *pin_capacitance* attribute on the Z pin of IV from the tech_lib library and sets that value on the input_1 and input_2 ports.

```
pt_shell> set_load \\
            [get_attribute [get_lib_pins tech_lib/IV/Z] pin_capacitance]
 \\
            [get_ports {input_1 input_2}]
```

The following example sets a capacitance of 3 on the U1/U2/NET3 net. The wire capacitance is set to 3. (Total net capacitance is 3 + the sum of the pin and port capacitances.)

```
pt_shell> set_load 3 U1/U2/NET3
```

The following example sets a total net capacitance (wire capacitance + pin capacitances) of 3 on the U1/U2/NET3 net. If the pin and port capacitances equal 2, the wire capacitance is annotated with 1; if the pin and port capacitances are 3 or greater, the wire capacitance is annotated with 0.

```
pt_shell> set_load -subtract_pin_load 3 U1/U2/NET3
```

The following example sets a wire capacitance of 5 units on the xyz port.

```
pt_shell> set_load -wire_load 5 xyz
```

The following example sets different capacitances values for different scenarios.

Feedback

```
pt_shell> set_load -sms_scenario $low_low 2 [get_port OUT]
pt_shell> set_load -sms_scenario $high_high 3 [get_port OUT]

pt-shell> get_attribute [get_port OUT] pin_capacitance_max($low_low)
2.0
pt-shell> get_attribute [get_port OUT] pin_capacitance_max($high_high)
3.0
```

The following example removes the back-annotated capacitance on a port and on a net.

```
pt_shell> remove_capacitance [get_ports the_answer]
pt_shell> remove_capacitance [get_nets net12]
```

See Also

- all_outputs
- current_design
- remove_capacitance
- report_net
- report_port
- reset_design
- set_drive
- set_wire_load_mode

---

set_max_area

Sets the *max_area* attribute on the current design to a specified value.

Syntax

int *set_max_area*

area_value

Data Types

area_value float

Feedback

Arguments

area_value

Specifies the value to which the *max_area* attribute is to be set. The value must be greater than or equal to 0 (>= 0). The units of the *area_value* option must be consistent with the units in the technology library used during optimization.

Description

The *set_max_area* command sets the *max_area* attribute on the current design to the specified *area_value*. This attribute represents the target area of the design.

Use the *remove_max_area* or *reset_design* command to remove the *max_area* attribute on the current design.

Use the *report_constraint* command to show area cost of the design.

Examples

The following example sets the target area for the current design to 250.

```
pt_shell> set_max_area 250.0
1
pt_shell> get_attribute [current_design] max_area
250.000000
```

See Also

- current_design
- get_attribute
- remove_max_area
- report_constraint
- reset_design

---

set_max_capacitance

Sets maximum capacitance limit for pins, library pins, ports, clocks, or designs.

Syntax

string *set_max_capacitance*

```
[-clock_path]
[-data_path]
[-rise]
[-fall]
[-force]
```

Feedback

```
[-sms_scenarios sms_scenarios_list]
capacitance_value
object_list
```

Data Types

```
capacitance_value          float
object_list                list
sms_scenarios_list         collection
```

Arguments

`-clock_path`

Applies the limit only to pins in the clock network of the clocks specified in the *object_list*.

`-data_path`

Applies the limit only to pins in the data paths launched by the clocks specified in the *object_list*.

`-rise`

Applies the limit only to rising capacitance values. This option requires that the *object_list* contains only clock objects.

`-fall`

Applies the limit only to falling capacitance values. This option requires that the *object_list* contains only clock objects.

`-force`

Force the limit to be set to the specified value, overriding other more restrictive constraints if they exist.

`-sms_scenarios` *sms_scenarios_list*

Limits the specification to only the given SMS scenarios. Without this option, the specification applies to all SMS scenarios. This collection is created by the *get_sms_scenarios* command.

capacitance_value

Specifies the capacitance limit (value >= 0). This is the maximum total capacitance (pin plus wire capacitance) in library capacitance units.

object_list

Provides a list of pins, library pins, ports, clocks, or designs on which to set maximum capacitance.

Feedback

Description

Specifies a maximum capacitance limit on pins, library pins, ports, clocks, or designs. If maximum capacitance is set on a pin or port, the net connected to that pin or port is expected to have a total capacitance less than the specified *capacitance_value*. If specified on a design, the default maximum capacitance for that design is set. If specified on a library pin, the requirement is inherited by all leaf pin instances.

If maximum capacitance is set on a clock, the maximum capacitance is applied to all pins in this specified clock domain (clock and data). Within a clock domain, you can optionally restrict the constraint further to only clock paths or data paths, and to only rising or falling capacitances. (Note that the *-clock_path*, *-data_path*, *-rise*, and *-fall* options can only be used for limits applied to clock objects.)

During analysis, the most restrictive of the design, pin, clock, library, and port limits is used.

The *report_constraint -max_capacitance* command shows maximum capacitance constraint evaluations. The minimum slack of rise and fall capacitance is reported. When there are violations across multiple clock-specific limit values, the worst slack is reported.

The *report_port -design_rule* command shows port maximum capacitance limits. The *report_design* command shows the default maximum capacitance limit for the current design.

To remove maximum capacitance limits applied by this command, use the *remove_max_capacitance* command.

Examples

The following example sets a maximum capacitance limit of 2.0 units on ports "OUT*".

```
pt_shell> set_max_capacitance 2.0 [get_ports "OUT*"]
```

The following example sets the default maximum capacitance limit of 5.0 units on the current design.

```
pt_shell> set_max_capacitance 5.0 [current_design]
```

The following example sets a maximum capacitance limit of 0.8 units on all pins in the clock network of CLK.

```
pt_shell> set_max_capacitance 0.8 [get_clocks CLK] -clock_path
```

The following example sets a maximum capacitance limit of 0.7 units on all pins of the data path of CLK.

```
pt_shell> set_max_capacitance 0.7 [get_clocks CLK] -data_path
```

Feedback

See Also

- current_design
- get_ports
- remove_max_capacitance
- report_constraint
- report_design
- report_port
- set_min_capacitance

---

set_max_delay

Specifies the maximum delay constraint for timing paths that start from, pass through, or end at specified objects.

Syntax

status *set_max_delay*

```
[-rise]
[-fall]
[-reset_path]
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-to to_list]
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-through through_list
   | -rise_through rise_through_list
   | -fall_through fall_through_list]
[-ignore_clock_latency]
[-probe]
[-comment comment_string]
[-sms_scenarios sms_scenarios_list]
delay_value
```

Data Types

comment_string	string
delay_value	float
fall_from_list	list
fall_through_list	list
fall_to_list	list
from_list	list
rise_from_list	list

```
rise_through_list          list
rise_to_list               list
sms_scenarios_list         collection
through_list               list
to_list                    list
```

Arguments

`-rise`

Applies the delay constraint only to paths with a rising transition at the path endpoint.

`-fall`

Applies the delay constraint only to paths with a falling transition at the path endpoint.

By default, the constraint applies to both rising and falling transitions.

`-reset_path`

Removes existing point-to-point timing exceptions from the specified paths before applying the maximum delay constraint. Using this option is equivalent to using the *reset_path* command with similar arguments before the *set_max_delay* command. Only exceptions defined with the same path settings are affected: *-from*, *-to*, *-rise*, *-fall*, *-setup*, *-hold*, and so on.

`-from` *`from_list`*

Specifies a list of timing path startpoint objects: a clock, primary input port, sequential cell, clock pin of a sequential cell, data pin of a level-sensitive latch, or pin that has an input delay specified.

If you specify a clock, all registers and primary inputs related to that clock are considered path startpoints. If you specify a sequential cell, the command is automatically expanded to apply to each individual clock input pin of the cell. If public variable *timing_enable_from_clock_to_dangling_load_endpoints* is set to true, then all registers and primary inputs related to the clock are used as startpoints to constrain any dangling load endpoints in their fanout.

You can specify any pin in a data path as a startpoint so that the *report_timing* command checks the path delay starting from that pin. This effectively breaks the data path into two segments and prevents path tracing through the pin. To prevent breaking of other timing paths through the pin, use the *-probe* option.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the delay constraint applies only to rising transitions at the specified objects.

If you specify a clock, the delay constraint applies to paths with a startpoint clocked by the named clock and launched by a rising edge at the clock source. The actual edge at the launch register can be either rising or falling, depending on logical inversions in the clock network.

`-fall_from` *`fall_from_list`*

Same as the *-rise_from* option, except that the constraint applies only to falling transitions at the specified objects.

You can use no more than one of the *-from*, *-rise_from*, and *-fall_from* options.

`-to` *`to_list`*

Specifies a list of timing path endpoint objects: a clock, primary output port, sequential cell, data pin of a sequential cell, or pin that has an output delay specified.

If you specify a clock, all registers and primary outputs related to that clock are considered path endpoints. If you specify a sequential cell, the command is automatically expanded to apply to each individual data input pin of the cell.

You can specify any pin in a data path as an endpoint so that the *report_timing* command checks the path delay ending at that pin. This effectively breaks the data path into two segments and prevents path tracing through the pin. To prevent breaking of other timing paths through the pin, use the *-probe* option.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, except that the delay constraint applies only to rising transitions at the specified objects.

If you specify a clock, the delay constraint applies to paths with an endpoint clocked by the named clock and captured by a rising edge at the clock source. The actual edge at the capture register can be either rising or falling, depending on logical inversions in the clock network.

`-fall_to` *`fall_to_list`*

Same as the *-rise_to* option, except that the constraint applies only to falling transitions at the specified objects.

You can use no more than one of the *-to*, *-rise_to*, and *-fall_to* options.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, and nets through which the paths must pass for the delay constraint to apply.

`-rise_through` *`rise_through_list`*

Same as the *-through* option, but applies only to paths with a rising transition at the specified objects.

`-fall_through` *`fall_through_list`*

Same as the *-through* option, but applies only to paths with a falling transition at the specified objects.

`-ignore_clock_latency`

Causes the launch and capture clock latencies to be ignored when the *report_timing* command computes slack for the specified paths; and if SI analysis is enabled, causes path-based reporting to conservatively use graph-based crosstalk delta delays. By default, launch and capture clock latencies are considered in the delay slack calculation, and path-based reporting uses path-based crosstalk delta delays.

`-probe`

Prevents the breaking of timing paths at a specified *-from* pin that is not the clock pin of a sequential device, or at a specified *-to* pin that is not the data input pin of a sequential device. Instead, the timing exception applies to the specified startpoints and endpoints without affecting other paths through the specified pins.

If you do not use the *-probe* option, specifying an intermediate pin of a path as a startpoint or endpoint forces that pin to be a startpoint or endpoint for all timing paths, effectively breaking all paths into two separate segments at that pin.

`-comment` *`comment_string`*

Associates a comment string with the command, which is included in the command text written out by the *write_sdc* command. This can be useful for tracking information about the source of the command.

`delay_value`

Specifies the maximum allowed delay for the path, in time units of the logic library used for timing analysis. For this maximum delay constraint, the clock period and clock edge times are ignored.

If a path startpoint has an input delay specified, that delay value is added to the path delay. If a path startpoint is on a sequential device, launch clock latency is included in the computed delay.

If the path endpoint has an output delay specified, that delay is used in the data required-time calculation. If a path endpoint is on a sequential device, the library setup time and capture clock latency are used in the data required-time calculation.

To prevent consideration of launch and capture clock latencies, use the *-ignore_clock_latency* option.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this maximum delay constraint is applied. When this option is not given the maximum delay constraint applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

This command specifies a maximum delay constraint for timing paths that start from, pass through, or end at specified objects in the design. This constraint overrides the default setup time constraint established by the arrival times of launch and capture clock edges.

The *report_timing* command reports violations of the maximum delay constraint, taking into account the input delay, output delay, library setup time at the endpoint register, and clock latencies. To prevent consideration of clock latencies, use the *-ignore_clock_latency* option of the *set_max_delay* command.

The *set_max_delay* command is a point-to-point timing exception command like the *set_multicycle_path*, *set_min_delay*, and *set_false_path* commands. A *set_max_delay* or *set_min_delay* command has higher priority than a *set_multicycle_path* command.

If you specify a sequential cell as the argument of the *-from*, *-rise_from*, or *-fall_from* option, the command is automatically expanded to apply to each clock input pin of the cell. If the cell has multiple clock input pins, the exceptions are maintained separately for each pin. You can see these exceptions by using the *report_exceptions* or *write_sdc* command.

Similarly, if you specify a sequential cell as the argument of the *-to*, *-rise_to*, or *-fall_to* option, the command is automatically expanded to apply to each data input pin of the cell.

The expansion of an exception set on a sequential cell to the individual pins of the cell does not occur in some other tools such as IC Compiler II. In those tools, the exception is maintained at the cell level.

Using the *-from* and *-to* (and similar) options of the *set_max_delay* command, you can specify any pin in a data path as a startpoint or endpoint so that the *report_timing* command checks the path delay between those points. The *set_max_delay* command responds with a UITE-217 warning message because path tracing stops at the specified point, effectively breaking the path into multiple segments. If you set a startpoint or endpoint in a clock network, clock propagation stops at that point. To remove a maximum delay constraint that has been set, use the *reset_path* command.

For a nonsequential timing check defined by the *set_data_check* command, the maximum delay constraint is valid only if clocked signals arrive at both the related ("from") pin and the constrained ("to") pin of the data check. If a clock cannot reach the related pin or constrained pin, the path is considered unconstrained and the timing check is not performed.

When multiple exceptions commands of different types apply to the same path, they apply with the following precedence (highest to lowest):

1. set_false_path
2. set_max_delay and set_min_delay
3. set_multicycle_path

When multiple exception commands of the same type apply to a path, the more specific command overrides the more general. They apply with the following precedence (more specific to more general):

1. set_max_delay -from *pin* -to *pin*
2. set_max_delay -from *pin* -to *clock*
3. set_max_delay -from *pin*
4. set_max_delay -from *clock* -to *pin*
5. set_max_delay -to *pin*
6. set_max_delay -from *clock* -to *clock*
7. set_max_delay -from *clock*
8. set_max_delay -to *clock*

When multiple exception commands of the same specificity apply that differ only by the presence or absence of the *-ignore_clock_latency* option, the command *without* the option takes precedence:

1. set_max_delay
2. set_max_delay -ignore_clock_latency

If multiple exceptions have the same path options and differ only in the specified rise/fall sense, the most constraining time value has priority. For example, the following commands are arranged in order, from the highest to lowest priority, due to their increasing maximum delay values.

1. set_max_delay -fall_from *pin* -to *pin* 5.0
2. set_max_delay -from *pin* -to *pin* 5.5
3. set_max_delay -rise_from *pin* -to *pin* 6.0

To report the current *set_max_delay* settings and other timing exception settings, use the *report_exceptions* command.

To remove timing exceptions set by the *set_max_delay* command, use the *reset_path* command with the same path settings; or the *reset_design* command to remove all timing constraints, including clocks.

Examples

The following example specifies that the delay can be no more than 3.0 time units for a path from a specific register clock pin to a specific output port. This new constraint overrides the default setup constraint for the path.

```
pt_shell> set_max_delay 3.0 -from I_ORCA_TOP/I_SDRAM/reg[8]/CPN -to
 sd_DQ[8]
```

```
Information: Invalidating logical update. (PTE-139)
1
pt_shell> report_timing -from I_ORCA_TOP/I_SDRAM/reg[8]/CPN -to sd_DQ[8]
 ...

  Point                                                   Incr       Path
  ------------------------------------------------------------------------
  clock network delay (propagated)                       2.281      2.281
  I_ORCA_TOP/I_SDRAM/reg[8]/CPN (sepfq1)                 0.000      2.281 f
  I_ORCA_TOP/I_SDRAM/reg[8]/Q (sepfq1)                   0.347      2.628 f
  I_ORCA_TOP/I_SDRAM/sd_mux_dq_out_8/Z (mx02d4)          0.715      3.343 f
  sdram_DQ_iopad_8/PAD (pc3b05)                          2.071      5.414 f
  sd_DQ[8] (inout)                                       0.048      5.461 f
  data arrival time                                                 5.461

  max_delay                                              3.000      3.000
  clock network delay (propagated)                       3.777      6.777
  clock reconvergence pessimism                          0.160      6.937
  output external delay                                 -0.750      6.187
  data required time                                                6.187
  ------------------------------------------------------------------------
  data required time                                                6.187
  data arrival time                                                -5.461
  ------------------------------------------------------------------------
  slack (MET)                                                       0.726
```

The timing check considers clock network delays (latencies) of the launch and capture clocks, but not the clock edge times.

To ignore the clock network delays as well as clock edge times, use the *set_max_delay* command with the *-ignore_clock_latency* option:

```
pt_shell> set_max_delay 3.0 -from I_ORCA_TOP/I_SDRAM/reg[8]/CPN -to
 sd_DQ[8] \\
           -ignore_clock_latency
Information: Invalidating logical update. (PTE-139)
1
pt_shell> report_timing -from I_ORCA_TOP/I_SDRAM/reg[8]/CPN -to sd_DQ[8]
 ...

  Point                                                   Incr       Path
  ------------------------------------------------------------------------
  I_ORCA_TOP/I_SDRAM/reg[8]/CPN (sepfq1)                 0.000      0.000 f
  I_ORCA_TOP/I_SDRAM/reg[8]/Q (sepfq1)                   0.347      0.347 f
  I_ORCA_TOP/I_SDRAM/sd_mux_dq_out_8/Z (mx02d4)          0.715      1.062 f
  sdram_DQ_iopad_8/PAD (pc3b05)                          2.071      3.132 f
  sd_DQ[8] (inout)                                       0.048      3.180 f
  data arrival time                                                 3.180

  max_delay                                              3.000      3.000
  clock reconvergence pessimism                          0.000      3.000
  output external delay                                 -0.750      2.250
```

```
  data required time                                                 2.250
  ------------------------------------------------------------------------
  data required time                                                 2.250
  data arrival time                                                 -3.180
  ------------------------------------------------------------------------
  slack (VIOLATED)                                                  -0.930
```

The following *reset_path* command removes the constraint set by the *set_max_delay* command, restoring the default setup timing check for the path.

```
pt_shell> reset_path -from I_ORCA_TOP/I_SDRAM/reg[8]/CPN -to sd_DQ[8]
1
pt_shell> report_timing -from I_ORCA_TOP/I_SDRAM/reg[8]/CPN -to sd_DQ[8]
 ...

  Point                                                 Incr       Path
  ------------------------------------------------------------------------
  clock SDRAM_CLK (fall edge)                          3.750      3.750
  clock network delay (propagated)                     2.281      6.031
  I_ORCA_TOP/I_SDRAM/reg[8]/CPN (sepfq1)               0.000      6.031 f
  I_ORCA_TOP/I_SDRAM/reg[8]/Q (sepfq1)                 0.347      6.378 f
  I_ORCA_TOP/I_SDRAM/sd_mux_dq_out_8/Z (mx02d4)        0.715      7.093 f
  sdram_DQ_iopad_8/PAD (pc3b05)                        2.071      9.164 f
  sd_DQ[8] (inout)                                     0.048      9.211 f
  data arrival time                                               9.211

  clock SD_DDR_CLK (rise edge)                         7.500      7.500
  clock network delay (propagated)                     3.777     11.277
  clock reconvergence pessimism                        0.160     11.437
  output external delay                               -0.750     10.687
  data required time                                             10.687
  ------------------------------------------------------------------------
  data required time                                             10.687
  data arrival time                                              -9.211
  ------------------------------------------------------------------------
  slack (MET)                                                     1.476
```

The following command specifies that all paths from flip-flop ff1a or ff1b to flip-flop ff2e must have a delay of no more than 6.0 time units.

```
pt_shell> set_max_delay -from {ff1a ff1b} -to {ff2e} 6.0
```

The following command specifies that all paths to endpoints clocked by PHI2 must have a delay of no more than 8.5 time units.

```
pt_shell> set_max_delay -to [get_clocks PHI2] 8.5
```

The following command specifies that all paths leading to ports named "busA[*]" must have a delay of no than 5.0 time units.

```
pt_shell> set_max_delay 5.0 -to "busA[*]"
```

The following command specifies that all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C}, in that order, must have a delay of no more than 8.0 time units.

```
pt_shell> set_max_delay 8.0 -from ff1/CP \\
            -through {U1/Z U2/Z} -through {U3/Z U4/C} -to ff2/D
```

The following command specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C}, in that order, must have a delay of no more than 8.0 time units.

```
pt_shell> set_max_delay 8.0 -from ff1/CP -rise_through {U1/Z U2/Z} \\
            -fall_through {U3/Z U4/C} -to ff2/D
```

See Also

- create_clock
- current_design
- group_path
- report_constraint
- report_path_group
- reset_design
- reset_path
- set_false_path
- set_input_delay
- set_min_delay
- set_multicycle_path
- set_output_delay
- timing_enable_from_clock_to_dangling_load_endpoints

---

set_max_fanout

Sets maximum fanout for input ports or designs.

Syntax

string *set_max_fanout*

Feedback

```
[-sms_scenarios sms_scenarios_list
fanout_value
object_list
```

Data Types

```
fanout_value          float
object_list           list
sms_scenarios_list    collection
```

Arguments

fanout_value

Fanout limit (Value >= 0). This is the maximum fanout load in library fanout units.

object_list

Provides a list of input ports or designs on which to set maximum fanout.

-sms_scenarios *sms_scenarios_list*

Specifies a collection of SMS scenarios for which this maximum fanout is applied. When this option is not given the maximum fanout applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

Specifies a maximum fanout on input ports or designs. If maximum fanout is set on a port, the net connected to that port is expected to have the *fanout_load* attribute less than the specified *fanout_value* attribute. Fanout load for a net is the sum of the *fanout_load* attributes for all input pins on the net. If specified on a design, the default maximum fanout for that design is set. Library cell pins may also have the *max_fanout* value specified. The most restrictive of the design limit and the pin or port limit will be used.

The *report_constraint -max_fanout* command shows maximum fanout constraint evaluations. The *report_port -design_rule* command shows port maximum fanout limits. The *report_design* command shows the default maximum fanout setting for the current design.

To remove maximum fanout limits from designs or ports, use the *remove_max_fanout* command.

Examples

The following example sets a maximum fanout limit of 2.0 units on ports "IN*".

```
pt_shell> set_max_fanout 2.0 [ge_ports "IN*"]
```

The following example sets the default maximum fanout limit of 5.0 units on the current design.

```
pt_shell> set_max_fanout 5.0 [current_design]
```

Feedback

See Also

- current_design
- remove_max_fanout
- report_constraint
- report_design
- report_port
- get_ports
- set_fanout_load

---

set_max_time_borrow

Limits time borrowing for latches.

Syntax

status *set_max_time_borrow*

```
value
object_list
```

Data Types

```
value          float
object_list    list
```

Arguments

value

Specifies the value to which the *max_time_borrow* attribute is set. Specifies the limit of time borrowing on the latches specified in the *object_list* option. The *delay_value* option must be between zero and the default maximum derived from the waveform. By default, the maximum is derived from the ideal clock waveform driving each latch, and is equal to (closing_edge - open_edge). Library setup and data-to-Q propagation times are automatically taken into account. The *delay_value* option is in the same units as those in the logic library used during analysis.

object_list

Specifies a list of objects in the *current_design* command for which time borrowing is to be limited to the *value* option. The objects can be clocks, latch cells, data pins, or clock (enable) pins. If a cell is specified, all enabled pins on that cell are affected.

Feedback

Description

Sets the *max_time_borrow* attribute to the *value* option to constrain the amount of time borrowing possible for level-sensitive latches. To meet delay targets, the *set_max_time_borrow* command prevents automatic use of all or part of the enabling clock pulse on a latch.

If the *max_time_borrow* attribute is set on both data and clock (enable) pins of a level-sensitive latch, the value set on the data pin takes higher priority.

To get the *max_time_borrow* attributes on the design, use the *get_attribute* or *report_exceptions* command.

To undo the *set_max_time_borrow* command, use the *remove_max_time_borrow* command.

Examples

The following example restricts time borrowing on latches *latch1a* and *latch2f* to 4.0 units.

```
pt_shell> set_max_time_borrow 4.0 { latch1a latch2f }
```

The following example specifies that no time borrowing will take place on *latch1c*.

```
pt_shell> set_max_time_borrow 0.0 { latch1c }
```

See Also

- current_design
- get_attribute
- remove_max_time_borrow
- report_exceptions

---

set_max_transition

Sets maximum transition limit for pins, ports, clocks, or designs with respect to the main library trip-points.

Syntax

string *set_max_transition*

```
[-clock_path]
[-data_path]
[-rise]
[-fall]
[-force]
[-sms_scenarios sms_scenarios_list]
```

transition_value
object_list

Data Types

object_list	list
sms_scenarios_list	collection
transition_value	float

Arguments

`-clock_path`

Applies the limit only to pins in the clock network of the clocks specified in the *object_list*.

`-data_path`

Applies the limit only to pins in the data paths launched by the clocks specified in the *object_list*.

`-rise`

Applies the limit only to rising transition values. This option requires that the *object_list* contains only clock objects.

`-fall`

Applies the limit only to falling transition values. This option requires that the *object_list* contains only clock objects.

`-force`

Force the limit to be set to the specified value, overriding other more restrictive constraints if they exist.

This option is supported only for pin and port objects.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios for which this maximum transition is applied. When this option is not given the maximum transition applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

transition_value

Specifies the transition limit (value >= 0). This is the maximum transition time in library time units.

object_list

Provides a list of pins, library pins, ports, clocks, or designs on which to set maximum transition.

Feedback

Description

Specifies a maximum transition limit on pins, library pins, ports, clocks, or designs. If maximum transition is set on a pin, library pin, or port, the pin, library pin, or port is expected to have transition time less than the specified *transition_value* option. If specified on a design, the default maximum transition for that design is set.

If maximum transition is set on a clock, the maximum transition is applied to all pins in this specified clock domain (clock and data). Within a clock domain, you can optionally restrict the constraint further to only clock paths or data paths, and to only rising or falling transitions. (Note that the *-clock_path*, *-data_path*, *-rise*, and *-fall* options can only be used for limits applied to clock objects.)

During analysis, the most restrictive of the design, pin, clock, library, and port limits is used. The limit value is scaled to the appropriate library pin trip-points as needed during delay calculation.

The *report_constraint -max_transition* command shows maximum transition constraint evaluations. The actual limit used for slack computation is reported. If a user-defined value is used, the reported limit is different for pins with different trip-points. The minimum slack of rise and fall transition is reported. When there are violations across multiple clock-specific limit values, the worst slack is reported.

The *report_port -design_rule* command shows port maximum transition limits. The *report_design* command shows the default maximum transition setting for the current design.

To remove maximum transition limits applied by this command, use the *remove_max_transition* command.

Examples

The following example sets a maximum transition limit of 2.0 units on ports "OUT*".

```
pt_shell> set_max_transition 2.0 [get_ports "OUT*"]
```

The following example sets the default maximum transition limit of 5.0 units on the current design.

```
pt_shell> set_max_transition 5.0 [current_design]
```

The following example sets a maximum transition limit of 2.0 units on all pins in the clock network of CLK.

```
pt_shell> set_max_transition 2.0 [get_clocks CLK] -clock_path
```

The following example sets a maximum transition limit of 1.8 units on all pins of the data path of CLK.

```
pt_shell> set_max_transition 1.8 [get_clocks CLK] -data_path
```

Feedback

See Also

- current_design
- get_ports
- remove_max_transition
- report_constraint
- report_design
- report_port

---

set_memory_cell_info

Specify memory library cells for RTL power analysis.

Syntax

int *set_memory_cell_info*

```
[-raddr pin_list]
[-waddr pin_list]
[-dpins pin_list]
[-qpins pin_list]
[-clock pin_name]
[-rclk pin_name]
[-wclk pin_name]
[-read  condition]
[-write condition]
[-disable condition]
libcell_list
```

Data Types

pin_list	list
pin_name	string
condition	string
libcell_list	list

Arguments

`-raddr`

Specifies the list of Read Address pins.

`-waddr`

Specifies the list of Write Address pins.

`-dpins`

Specifies the list of Input Data pins.

Feedback

`-qpins`

Specifies the list of Output pins.

`-clock`

Specifies the Clock pin.

`-rclk`

Specifies the Read Clock pin.

`-wclk`

Specifies the Write Clock pin.

`-read`

Specifies a boolean condition to indicate when the memory performs read operation.

`-write`

Specifies a boolean condition to indicate when the memory performs write operation.

`-disable`

Specifies a boolean condition to indicate when the memory is disabled.

`libcell_list`

Specifies a list of memory library cells or a single reference name.

Description

The *set_memory_cell_info* command is used to identify memory library cells and their read/write/input/output/clock pins. This is useful for various applications such as reporting the power and attributes of memory instances in the design.

Examples

The following example shows how to use this command.

```
pwr_shell> set_memory_cell_info -raddr RA[*] -waddr WA[*] -qpins
 DO[*] -dpins DI[*] -read "!CS & WE" -write "!CS & !WE" -clock CLK
 [get_lib_cells */sram32x64_hscm4]

pwr_shell> set_memory_cell_info -raddr AB[*] -waddr AA[*] -dpins D[*]
 -qpins Q[*] -read !REB -write !WEB -disable "REB & WEB" -rclk CLKR -wclk
 CLKW [get_lib_cells */FFCLLSVTA80X18M2SSHO]
```

See Also

- report_rtl_metrics

Feedback

set_memory_percentage

Sets percentage of memory cells to be switched during vector free vector generation.

Syntax

status *set_memory_percentage*

```
-value percentage_value
[-type type]
```

Data Types

```
percentage_value  float
type              type
```

Arguments

`-value percentage_value`

Specifies percentage of memory cells which will be switched during vector generation. Accepted value is between 0.0 and 1.0, which means 0% to 100% of memory cells have switching events triggered. When this option is specified, it is used to guide vector generation such that the percentage of memory cells to be switched does not exceed this limit. Default value is 0.5, which means 50% memory cells will have switching events, and the other 50% memory cells are in steady state.

`-type type`

Specifies the type of memory cells for which the percentage value is applied. Accepted values are icg_controlled and non_icg_controlled. The type *icg_controlled* means the percentage value is applied to memory cells where clocks are controlled by clock gating cells. The type *non_icg_controlled* means the percentage value is applied to memory cells where clocks are controlled by non clock gating cells, for example, MUX cells. When this option is omitted, the percentage value is applied to all memory cells in the whole design.

Description

This command is used to specify percentage of memory cells to be switched during vector generation in command *write_vectors*.

Examples

The following example sets memory cell percentage 20% instead of default 50%, then invokes the command *write_vectors*.

```
pt_shell> set_memory_percentage -value 0.2
pt_shell> write_vectors rtl.vcd
```

Feedback

Assuming the whole design consists of two types of memory cells. The following example sets 30% for memory cells controlled by clock gating cells, and 10% for memory cells controlled by non clock gating cells. Then the command *write_vectors* is invoked:

```
pt_shell> set_memory_percentage -value 0.3 -type icg_controlled
pt_shell> set_memory_percentage -value 0.1 -type non_icg_controlled
pt_shell> write_vectors rtl.vcd
```

See Also

- write_vectors

---

set_message_info

Sets limits on the numbers of tool messages issued, stores messages for retrieval, or forces a Tcl error on specified system conditions.

Syntax

string *set_message_info*

```
-id message_id ]
[-limit max_limit]
[-save_limit max_limit]
[-stop_on]
[-stop_off]
```

Data Types

```
message_id   string
max_limit    integer
```

Arguments

`-id message_id`

Specifies the tool message ID affected by the command, for example, RC-011 or UITE-214.

`-limit max_limit`

Sets the maximum number of times that the message is reported in the remainder of the current session. When the limit is reached, the tool issues a warning that future messages of that type will be suppressed. After that, occurrences of the related condition no longer generate messages.

Setting this option to 0 means no limit on the number of messages generated.

`-save_limit` *`max_limit`*

Sets the maximum number of times that the message is saved for later retrieval by the *get_attribute* command or query by the *get_message_info* command.

Setting this option to 0 means no limit on the number of messages saved for later retrieval. Setting this option to -1 means no messages are saved, which is the default.

`-stop_on`

Causes any occurrence of the specified message ID to trigger a Tcl error, which stops execution of a Tcl script being run.

`-stop_off`

Cancels the effects of the *-stop_on* option, so that occurrences of the specified message ID no longer stop execution of a Tcl script.

Description

The *set_message_info* command lets you do any one of the following:

- Set a limit on the number of tool messages (error, warning, or information) of a specified message ID type generated in the remainder of the session; messages beyond this limit are suppressed.
- Save system messages of a specified type for later retrieval by the *get_message_info* and *get_attribute* commands.
- Cause the condition associated with a specific message to generate a Tcl error, which stops execution of the Tcl script that triggered the message.

To find out about the message settings and the number of messages that have been suppressed or saved, use the *get_message_info* command.

Examples

The following example sets a limit of 100 on the number of RC-011 messages reported in the remainder of the session. When the 100th RC-011 message is generated, the tool issues a warning that further messages will be suppressed, and then RC-011 conditions are no longer reported.

```
pt_shell> set_message_info -id RC-011 -limit 100
1
pt_shell> update_timing
 ...
Warning: An extrapolation far outside the library characterization range
 has
occurred for the cell timing arc, (my_lib/BUFD3) U564/I-->Z ...
 ...
 ...
```

```
Note - message 'RC-011' limit (100) exceeded.  Remainder will be
 suppressed.
 ...
```

The following example saves up to 100 PTE-064 messages for later retrieval.

```
pt_shell> set_message_info -id PTE-064 -save_limit 100
1
pt_shell> get_message_info -id PTE-064
id PTE-064 severity Information limit 10000 save_limit 100 occurrences
22 suppressed 0 message {Related clock set %d includes clock '%s' with
 period %.3f.}
 ...
pt_shell> set my_msgs [get_message_instances PTE-064]
_sel122
pt_shell> get_attribute -class message_instance $my_msgs arguments
0,SDRAM_CLK,7.500 0,SD_DDR_CLK,7.500 1,SYS_CLK,8.000 ...
 ...
pt_shell> get_attribute -class message_instance $my_msgs message_body
{Related clock set 0 includes clock 'SDRAM_CLK' with period 7.500.}
{Related clock set 0 includes clock 'SD_DDR_CLK' with period 7.500.}
pt_shell> get_attribute -class message_instance $my_msgs message_id
PTE-064 PTE-064 PTE-064 PTE-064 PTE-064 PTE-064 ...
```

The following example causes any occurrence of the PTE-023 warning condtion to trigger a Tcl error, which stops execution of the currenly running Tcl script.

```
pt_shell> set_message_info -id PTE-023 -stop_on
1
pt_shell> source my_script.tcl
 ...
Error: Severe error encountered
        Use error_info for more info. (CMD-013)
Information: script 'my_script.tcl'
        stopped at line 49 due to error. (CMD-081)
 ...
```

See Also

- get_attribute
- get_message_info
- get_message_ids
- print_message_info
- suppress_message

Feedback

set_min_capacitance

Sets minimum capacitance limit for ports or designs.

Syntax

string *set_min_capacitance*

```
[-sms_scenarios sms_scenarios_list]
capacitance_value
object_list
```

Data Types

capacitance_value	float
object_list	list
sms_scenarios_list	collection

Arguments

`-sms_scenarios sms_scenarios_list`

Specifies a collection of SMS scenarios for which this minimum capacitance is applied. When this option is not given the minimum capacitance applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

capacitance_value

Specifies the capacitance limit (value >= 0). This is the minimum total capacitance (pin plus wire capacitance) in library capacitance units.

object_list

Specifies a list of input or inout ports or designs on which to set minimum capacitance.

Description

Specifies a minimum capacitance on input/inout ports or designs. If minimum capacitance is set on a port, the net connected to that port is expected to have a total capacitance greater than the specified *capacitance_value*. If specified on a design, the default minimum capacitance for that design is set. During analysis, the most restrictive of the design and port limits is used. The *report_constraint -min_capacitance* command shows minimum capacitance constraint evaluations.

The *report_port -design_rule* command shows port minimum capacitance limits. The *report_design* command shows the default minimum capacitance limit for the current design.

To remove minimum capacitance limits applied by this command, use the *remove_min_capacitance* command.

Feedback

Examples

The following example sets a minimum capacitance limit of 0.2 units on ports "IN*".

```
pt_shell> set_min_capacitance 0.2 [get_ports "IN*"]
```

The following example sets the default minimum capacitance limit of 0.1 units on the current design.

```
pt_shell> set_min_capacitance 0.1 [current_design]
```

See Also

- current_design
- get_ports
- remove_min_capacitance
- report_constraint
- report_design
- report_port

---

set_min_delay

Specifies the minimum delay constraint for timing paths that start from, pass through, or end at specified objects.

Syntax

status *set_min_delay*

```
[-rise]
[-fall]
[-reset_path]
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-through through_list
   | -rise_through rise_through_list
   | -fall_through fall_through_list]
[-ignore_clock_latency]
[-probe]
[-comment comment_string]
[-sms_scenarios sms_scenarios_list]
delay_value
```

Feedback

Data Types

comment_string	string
delay_value	float
fall_from_list	list
fall_through_list	list
fall_to_list	list
from_list	list
rise_from_list	list
rise_through_list	list
rise_to_list	list
sms_scenarios_list	collection
through_list	list
to_list	list

Arguments

`-rise`

Applies the delay constraint only to paths with a rising transition at the path endpoint.

`-fall`

Applies the delay constraint only to paths with a falling transition at the path endpoint.

By default, the constraint applies to both rising and falling transitions.

`-reset_path`

Removes existing point-to-point timing exceptions from the specified paths before applying the minimum delay constraint. Using this option is equivalent to using the *reset_path* command with similar arguments before the *set_min_delay* command. Only exceptions defined with the same path settings are affected: *-from*, *-to*, *-rise*, *-fall*, *-setup*, *-hold*, and so on.

`-from` *from_list*

Specifies a list of timing path startpoint objects: a clock, primary input port, sequential cell, clock pin of a sequential cell, data pin of a level-sensitive latch, or pin that has an input delay specified.

If you specify a clock, all registers and primary inputs related to that clock are considered path startpoints. If you specify a sequential cell, the command is automatically expanded to apply to each individual clock input pin of the cell. If public variable *timing_enable_from_clock_to_dangling_load_endpoints* is set to true, then all registers and primary inputs related to the clock are used as startpoints to constrain any dangling load endpoints in their fanout.

You can specify any pin in a data path as a startpoint so that the *report_timing* command checks the path delay starting from that pin. This effectively breaks

the data path into two segments and prevents path tracing through the pin. To prevent breaking of other timing paths through the pin, use the *-probe* option.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the delay constraint applies only to rising transitions at the specified objects.

If you specify a clock, the delay constraint applies to paths with a startpoint clocked by the named clock and launched by a rising edge at the clock source. The actual edge at the launch register can be either rising or falling, depending on logical inversions in the clock network.

`-fall_from` *`fall_from_list`*

Same as the *-rise_from* option, except that the constraint applies only to falling transitions at the specified objects.

You can use no more than one of the *-from*, *-rise_from*, and *-fall_from* options.

`-to` *`to_list`*

Specifies a list of timing path endpoint objects: a clock, primary output port, sequential cell, data pin of a sequential cell, or pin that has an output delay specified.

If you specify a clock, all registers and primary outputs related to that clock are considered path endpoints. If you specify a sequential cell, the command is automatically expanded to apply to each individual data input pin of the cell.

You can specify any pin in a data path as an endpoint so that the *report_timing* command checks the path delay ending at that pin. This effectively breaks the data path into two segments and prevents path tracing through the pin. To prevent breaking of other timing paths through the pin, use the *-probe* option.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, except that the delay constraint applies only to rising transitions at the specified objects.

If you specify a clock, the delay constraint applies to paths with an endpoint clocked by the named clock and captured by a rising edge at the clock source. The actual edge at the capture register can be either rising or falling, depending on logical inversions in the clock network.

`-fall_to` *`fall_to_list`*

Same as the *-rise_to* option, except that the constraint applies only to falling transitions at the specified objects.

You can use no more than one of the *-to*, *-rise_to*, and *-fall_to* options.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, and nets through which the paths must pass for the delay constraint to apply.

`-rise_through` *`rise_through_list`*

Same as the *-through* option, but applies only to paths with a rising transition at the specified objects.

`-fall_through` *`fall_through_list`*

Same as the *-through* option, but applies only to paths with a falling transition at the specified objects.

`-ignore_clock_latency`

Causes the launch and capture clock latencies to be ignored when the *report_timing* command computes slack for the specified paths; and if SI analysis is enabled, causes path-based reporting to conservatively use graph-based crosstalk delta delays. By default, launch and capture clock latencies are considered in the delay slack calculation, and path-based reporting uses path-based crosstalk delta delays.

`-probe`

Prevents the breaking of timing paths at a specified *-from* pin that is not the clock pin of a sequential device, or at a specified *-to* pin that is not the data input pin of a sequential device. Instead, the timing exception applies to the specified startpoints and endpoints without affecting other paths through the specified pins.

If you do not use the *-probe* option, specifying an intermediate pin of a path as a startpoint or endpoint forces that pin to be a startpoint or endpoint for all timing paths, effectively breaking all paths into two separate segments at that pin.

`-comment` *`comment_string`*

Associates a comment string with the command, which is included in the command text written out by the *write_sdc* command. This can be useful for tracking information about the source of the command.

`delay_value`

Specifies the minimum allowed delay for the path, in time units of the logic library used for timing analysis. For this minimum delay constraint, the clock period and clock edge times are ignored.

If a path startpoint has an input delay specified, that delay value is added to the path delay. If a path startpoint is on a sequential device, launch clock latency is included in the computed delay.

If the path endpoint has an output delay specified, that delay is used in the data required-time calculation. If a path endpoint is on a sequential device, the library hold time and capture clock latency are used in the data required-time calculation.

To prevent consideration of launch and capture clock latencies, use the *-ignore_clock_latency* option.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this minimum delay constraint is applied. When this option is not given the minimum delay constraint applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

This command specifies a minimum delay constraint for timing paths that start from, pass through, or end at specified objects in the design. This constraint overrides the default hold time constraint established by the arrival times of launch and capture clock edges.

The *report_timing* command reports violations of the minimum delay constraint, taking into account the input delay, output delay, library hold time at the endpoint register, and clock latencies. To prevent consideration of clock latencies, use the *-ignore_clock_latency* option of the *set_min_delay* command.

The *set_min_delay* command is a point-to-point timing exception command like the *set_multicycle_path*, *set_max_delay*, and *set_false_path* commands. A *set_max_delay* or *set_min_delay* command has higher priority than a *set_multicycle_path* command.

If you specify a sequential cell as the argument of the *-from*, *-rise_from*, or *-fall_from* option, the command is automatically expanded to apply to each clock input pin of the cell. If the cell has multiple clock input pins, the exceptions are maintained separately for each pin. You can see these exceptions by using the *report_exceptions* or *write_sdc* command.

Similarly, if you specify a sequential cell as the argument of the *-to*, *-rise_to*, or *-fall_to* option, the command is automatically expanded to apply to each data input pin of the cell.

The expansion of an exception set on a sequential cell to the individual pins of the cell does not occur in some other tools such as IC Compiler II. In those tools, the exception is maintained at the cell level.

Using the *-from* and *-to* (and similar) options of the *set_min_delay* command, you can specify any pin in a data path as a startpoint or endpoint so that the *report_timing* command checks the path delay between those points. The *set_min_delay* command responds with a UITE-217 warning message because path tracing stops at the specified point, effectively breaking the path into multiple segments. If you set a startpoint or endpoint in a clock network, clock propagation stops at that point. To remove a minimum delay constraint that has been set, use the *reset_path* command.

For a nonsequential timing check defined by the *set_data_check* command, the minimum delay constraint is valid only if clocked signals arrive at both the related ("from") pin and the constrained ("to") pin of the data check. If a clock cannot reach the related pin or constrained pin, the path is considered unconstrained and the timing check is not performed.

When multiple exceptions commands of different types apply to the same path, they apply with the following precedence (highest to lowest):

```
1. set_false_path
2. set_max_delay and set_min_delay
3. set_multicycle_path
```

When multiple exception commands of the same type apply to a path, the more specific command overrides the more general. They apply with the following precedence (more specific to more general):

```
1. set_min_delay -from pin -to pin
2. set_min_delay -from pin -to clock
3. set_min_delay -from pin
4. set_min_delay -from clock -to pin
5. set_min_delay -to pin
6. set_min_delay -from clock -to clock
7. set_min_delay -from clock
8. set_min_delay -to clock
```

When multiple exception commands of the same specificity apply that differ only by the presence or absence of the *-ignore_clock_latency* option, the command *without* the option takes precedence:

```
1. set_min_delay
2. set_min_delay -ignore_clock_latency
```

If multiple exceptions have the same path options and differ only in the specified rise/fall sense, the most constraining time value has priority. For example, the following commands are arranged in order, from the highest to lowest priority, due to their decreasing minimum delay values.

```
1. set_min_delay -fall_from pin -to pin 2.5
2. set_min_delay      -from pin -to pin 2.0
3. set_min_delay -rise_from pin -to pin 1.5
```

To report the current *set_min_delay* settings and other timing exception settings, use the *report_exceptions* command.

To remove timing exceptions set by the *set_min_delay* command, use the *reset_path* command with the same path settings; or the *reset_design* command to remove all timing constraints, including clocks.

Feedback

Examples

The following example specifies that the delay must be at least 0.5 time units for a path from a specific input port to a specific register data pin. This new constraint overrides the default hold constraint for the path.

```
pt_shell> set_min_delay 0.5 -from sd_DQ[15] -to
 I_ORCA_TOP/I_SDRAM/reg[15]/D
Information: Invalidating logical update. (PTE-139)
1
pt_shell> report_timing -delay_type min \\
            -from sd_DQ[15] -to I_ORCA_TOP/I_SDRAM/reg[15]/D
 ...

  Point                                                 Incr       Path
  ---------------------------------------------------------------------
  input external delay                                 0.200      0.200 f
  sd_DQ[15] (inout)                                    0.000      0.200 f
  sdram_DQ_iopad_15/PAD (pc3b05)                       0.043      0.243 f
  sdram_DQ_iopad_15/CIN (pc3b05)                       0.467      0.710 f
  I_ORCA_TOP/I_SDRAM/Inst938/Z (buffd1)                0.092      0.802 f
  I_ORCA_TOP/I_SDRAM/Inst1079/Z (bufbd1)               0.092      0.894 f
  I_ORCA_TOP/I_SDRAM/reg[15]/D (sdnrb1)                0.000      0.894 f
  data arrival time                                               0.894

  min_delay                                            0.500      0.500
  clock network delay (propagated)                     2.102      2.602
  clock reconvergence pessimism                        0.000      2.602
  library hold time                                   -0.062      2.540
  data required time                                              2.540
  ---------------------------------------------------------------------
  data required time                                              2.540
  data arrival time                                              -0.894
  ---------------------------------------------------------------------
  slack (VIOLATED)                                               -1.646
```

The timing check considers clock network delays (latencies) of the launch and capture clocks, but not the clock edge times.

To ignore the clock network delays as well as clock edge times, use the *set_min_delay* command with the *-ignore_clock_latency* option:

```
pt_shell> set_min_delay 0.5 -from sd_DQ[15] -to
 I_ORCA_TOP/I_SDRAM/reg[15]/D \\
            -ignore_clock_latency
Information: Invalidating logical update. (PTE-139)
1
pt_shell> report_timing -delay_type min \\
            -from sd_DQ[15] -to I_ORCA_TOP/I_SDRAM/reg[15]/D
 ...

  Point                                                 Incr       Path
```

```
  ------------------------------------------------------------------
  input external delay                          0.200      0.200 f
  sd_DQ[15] (inout)                             0.000      0.200 f
  sdram_DQ_iopad_15/PAD (pc3b05)                0.043      0.243 f
  sdram_DQ_iopad_15/CIN (pc3b05)                0.467      0.710 f
  I_ORCA_TOP/I_SDRAM/Inst938/Z (buffd1)         0.092      0.802 f
  I_ORCA_TOP/I_SDRAM/Inst1079/Z (bufbd1)        0.092      0.894 f
  I_ORCA_TOP/I_SDRAM/reg[15]/D (sdnrb1)         0.000      0.894 f
  data arrival time                                        0.894

  min_delay                                     0.500      0.500
  clock reconvergence pessimism                 0.000      0.500
  library hold time                            -0.062      0.438
  data required time                                       0.438
  ------------------------------------------------------------------
  data required time                                       0.438
  data arrival time                                       -0.894
  ------------------------------------------------------------------
  slack (MET)                                              0.456
```

The following *reset_path* command removes the constraint set by the *set_min_delay* command, restoring the default hold timing check for the path.

```
pt_shell> reset_path -from I_ORCA_TOP/I_SDRAM/reg[8]/CPN -to sd_DQ[8]
1
pt_shell> report_timing -delay_type min \\
            -from sd_DQ[15] -to I_ORCA_TOP/I_SDRAM/reg[15]/D
 ...

  Point                                          Incr       Path
  ------------------------------------------------------------------
  clock SDRAM_CLK (rise edge)                   0.000      0.000
  clock network delay (propagated)              0.000      0.000
  input external delay                          0.200      0.200 f
  sd_DQ[15] (inout)                             0.000      0.200 f
  sdram_DQ_iopad_15/PAD (pc3b05)                0.043      0.243 f
  sdram_DQ_iopad_15/CIN (pc3b05)                0.467      0.710 f
  I_ORCA_TOP/I_SDRAM/Inst938/Z (buffd1)         0.092      0.802 f
  I_ORCA_TOP/I_SDRAM/Inst1079/Z (bufbd1)        0.092      0.894 f
  I_ORCA_TOP/I_SDRAM/reg[15]/D (sdnrb1)         0.000      0.894 f
  data arrival time                                        0.894

  clock SDRAM_CLK (rise edge)                   0.000      0.000
  clock network delay (propagated)              2.102      2.102
  clock reconvergence pessimism                 0.000      2.102
  I_ORCA_TOP/I_SDRAM/reg[15]/CP (sdnrb1)                   2.102 r
  library hold time                            -0.062      2.040
  data required time                                       2.040
  ------------------------------------------------------------------
  data required time                                       2.040
  data arrival time                                       -0.894
  ------------------------------------------------------------------
  slack (VIOLATED)                                        -1.146
```

The following command specifies that all paths from flip-flop ff1a or ff1b to flip-flop ff2e must have a delay of at least 1.5 time units.

```
pt_shell> set_min_delay -from {ff1a ff1b} -to {ff2e} 1.5
```

The following command specifies that all paths to endpoints clocked by PHI2 must have a delay of at least 1.2 time units.

```
pt_shell> set_min_delay -to [get_clocks PHI2] 1.2
```

The following command specifies that all paths leading to ports named "busA[*]" must have a delay of at least 1.8 time units.

```
pt_shell> set_min_delay 1.8 -to "busA[*]"
```

The following command specifies that all timing paths from ff1/CP to ff2/D that pass through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C}, in that order, must have a delay of at least 1.2 time units.

```
pt_shell> set_min_delay 1.2 -from ff1/CP \\
            -through {U1/Z U2/Z} -through {U3/Z U4/C} -to ff2/D
```

The following command specifies that all timing paths from ff1/CP to ff2/D that rise through one or more of {U1/Z U2/Z} and fall through one or more of {U3/Z U4/C}, in that order, must have a delay of at least 1.3 time units.

```
pt_shell> set_min_delay 1.3 -from ff1/CP -rise_through {U1/Z U2/Z} \\
            -fall_through {U3/Z U4/C} -to ff2/D
```

See Also

- create_clock
- current_design
- group_path
- report_constraint
- report_path_group
- reset_design
- reset_path
- set_false_path
- set_input_delay
- set_max_delay
- set_multicycle_path

Feedback

- set_output_delay
- timing_enable_from_clock_to_dangling_load_endpoints

---

set_min_library

Sets the library to be used for minimum delay analysis

The *set_min_library* command is used to relate a minimum conditions library to a maximum conditions library.

Syntax

string *set_min_library*

```
[-min_version min_library]
[-none]
max_library
```

Data Types

```
min_library     string
max_library     string
```

Arguments

`-min_version` *min_library*

The library for min analysis. This library is not to be used in the *link_path* command.

`-none`

Dissociates the *max_library* option from its min library.

max_library

The library for max analysis. This library should be used in the *link_path* command.

Description

The *set_min_library* command creates a max/min relationship between two libraries. After you establish the relationship, the tool uses the *max_library* for maximum delay analysis and the *min_library* for minimum delay analysis.

For each library cell in the *max_library*, the *set_min_library* command searches for a library cell of the same name in the *min_library*. If it is not found, a warning is issued, and there is no max/min relationship for that library cell. If the *min_library* does have the named library cell, the command compares the two library cells to verify that they have the same pins (in the same order, with the same directions), and the same timing arcs. If any

of these conditions fails, a warning is issued, and there is no max/min relationship for that library cell.

At least one library cell must match for the command to succeed.

When PrimeTime needs to compute a minimum delay value, if there is a max/min relationship for the library cell, the timing information from the min library is used. Otherwise, the information is taken from the max library.

Both libraries and objects in them can be referenced in commands such as the *set_operating_conditions* and *set_wire_load_model* commands.

Libraries used as the *min_library* in a *set_min_library* command should never be placed in the link path. Only the *max_library*, which is the root of the relationship, is used in the link path. A warning is issued if you use a min library in the link path.

A library in use as a min library can be removed with the *remove_lib* command as long as no environment information (operating conditions, wire load models, and so on) is being used from that library. When you remove a library in this way, any designs using the max/min library is scheduled for a full timing update.

The *list_libs* command shows max/min relationships. In addition, the *report_lib* command indicates when a max library has a relationship to a min library. See the examples or the man pages for these commands.

Examples

The following example shows a typical usage of *set_min_library*:

```
pt_shell> set_min_library LIB_WC_COM.db -min_version LIB_BC_COM.db
Loading db file '/u/libraries/LIB_WC_COM.db'
Loading db file '/u/libraries/LIB_BC_COM.db'
Created max/min library relationship:
  Max: /u/libraries/LIB_WC_COM.db:LIB_WC_COM
  Max: /u/libraries/LIB_BC_COM.db:LIB_BC_COM
1
pt_shell> list_libs
Library Registry:
   m LIB_BC_COM    /u/libraries/LIB_BC_COM.db:LIB_BC_COM
  *M LIB_WC_COM    /u/libraries/LIB_WC_COM.db:LIB_WC_COM

1
pt_shell> set_operating_conditions -max WC_COM -max_library LIB_WC_COM \\
          -min BC_COM -min_library LIB_BC_COM
1
pt_shell> report_timing -delay min
 ...
```

Feedback

See Also

- list_libs
- remove_lib
- report_lib
- set_operating_conditions
- set_wire_load_model
- link_path

---

set_min_pulse_width

Sets a minimum pulse width constraint for specified design objects.

Syntax

string *set_min_pulse_width*

```
[-low]
[-high]
value
[object_list]
```

Data Types

value	float
object_list	list

Arguments

`-low`

Indicates that the minimum pulse width constraint specified by *value* is to apply only to low clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

`-high`

Indicates that the minimum pulse width constraint specified by *value* is to apply only to high clock signal levels. If you do not specify the *-low* or *-high* option, both low and high pulses of clock signals are constrained.

value

A positive floating point value that specifies the minimum pulse width check to be applied to clock signals.

object_list

Specifies a list of clocks, cells, pins, or ports in the current design, to which the minimum pulse width check is to be applied. If you specify a cell or clock, all pins on the cell or clock are affected. If you do not specify any objects, the minimum pulse width check is applied to the current design; this value is applied to clock tree pulse width checks only (i.e. not sequential checks) if no pin-specific check (whether library- or user-specified) applies to a given pin.

Description

The *set_min_pulse_width* command specifies a minimum pulse width check to be applied to clock signals in the clock tree or at sequential devices. Use this command to add a more restrictive value than the one specified in library.

A clock pulse width problem occurs if the negation of the clock signal happens too soon after the assertion of the clock signal. Clock pulse width violations can cause the following problems:

- Sequential devices (flip-flops and latches) might not capture data properly.
- In some logic circuits, the entire pulse could disappear.

Two types of minimum pulse width checks are performed:

- Clock Pulse Width Check at Sequential Devices: This check is performed on clock pins of sequential elements. This check verifies that a minimum separation exists between the trailing edge and leading edge of the clock that is clocking the sequential elements. The library defines the constraints, which are environmentally scalable. The most restrictive value of pulse width (library-specified or user-asserted) is used. No default value is assumed.
- Clock Tree Pulse Width Check at Logic Circuits: The clock tree pulse width check ensures that the clock pulse is propagated reliably through the logic. This check is performed on the combinational logic circuits through which the clock signals are propagated. No default is assumed for the clock tree pulse width check. The most restrictive value of pulse width (library-specified or user-asserted) is used.

Note: If the *timing_enable_pulse_clock_constraints* variable is set to its default of *true*, the constraint set by this command does not apply to pulse clock networks. To apply the constraints to pulse clock networks use the *set_pulse_clock_min_width* command. To constrain pulse clock network using the *set_min_pulse_width* command from UI, set the *timing_enable_pulse_clock_constraints* variable to *false*.

To generate a report of pulse width constraints, use the *report_constraint -min_pulse_width* or *report_min_pulse_width* command. To remove minimum pulse width constraints set by *set_min_pulse_width*, use the *remove_min_pulse_width* command.

The *reset_design* command removes all user-specified attributes from a design, including those set by the *set_min_pulse_width* command.

Examples

The following example sets a minimum pulse width requirement of 2.0 for both low and high pulses of clock signals.

```
pt_shell> set_min_pulse_width 2.0 [get_clocks CK1]
```

The following example sets a minimum pulse width requirement of 2.5 for the low pulse of the clock signal on the pin U1/Z.

```
pt_shell> set_min_pulse_width -low 2.5 U1/Z
```

See Also

- current_design
- remove_min_pulse_width
- report_constraint
- report_min_pulse_width
- report_pulse_clock_min_width
- reset_design
- set_pulse_clock_min_width
- timing_enable_pulse_clock_constraints

---

set_mode

Selects the active mode of cell mode groups or design mode groups

Syntax

status *set_mode*

```
[-type cell | design]
[mode_list]
[instance_list]
```

Data Types

mode_list	list
instance_list	list

Feedback

Arguments

`-type cell | design`

Indicates the type of mode to be made active. This option has the following mutually-exclusive valid values: *design* and *cell*. The *cell* value specifies that cell modes are to be made active. Cell modes are defined on library cells in the library. The *design* value specifies that design modes are to be made active. Design modes are user-specified using the *define_design_mode_group* and *map_design_mode* commands. If the *-type* value is omitted from the command, then this is equivalent to specifying the *-type cell* value.

`mode_list`

Specifies a list of modes, each of which is to be made the active mode for its mode group. If the *-type* has a cell value, then the *mode_list* option must contain only cell modes. If the *-type* has a design value, then the mode list must contain only design modes.

`instance_list`

Specifies a list of instances for which the specified cell modes are made active. This list must only be included with the *-type cell* value.

Description

Selects the active mode for a mode group or for several mode groups and disables modes in the same group as the selected active mode. Mode groups must either have all modes enabled (default setting) or have one of their modes enabled and all others disabled. This command sets either cell modes or design modes depending on the value of the type option.

Cell mode groups are defined in the library. Each library cell can have a set of cell mode groups. Each of these cell mode groups can have two or more cell modes. Each of these cell modes can be mapped to a set of timing arcs of the library cell. When a cell mode is made active for a given instance of the library cell, the cell mode is enabled and all of its timing arcs are enabled for that cell. All other cell modes are automatically disabled. The timing arcs of the disabled cell modes are then automatically disabled. To view what modes have been enabled or disabled use the *report_mode -type* cell. To view what arcs have been disabled, use the *report_disable_timing* command.

In the special case of an arc having multiple cell modes mapped to it, the arc is enabled if any of the modes are enabled.

Cell modes can be made active in three different ways, using the *set_mode -type cell* command, using the *set_mode -type design* command or through evaluation of mode conditions. When conflict arises the following precedence rule is employed: *set_mode* -type cell > *set_mode* -type_design > evaluation of mode conditions

Feedback

Design modes and design mode groups are defined by you with the *define_design_mode_group* and *map_design_mode* commands. Design modes can be mapped to a set of cell modes or a set of paths. When a design mode of a particular design mode group is made active, all cell modes mapped to the design mode are made active and all paths mapped to the design mode are reset. All other design modes in the design mode group are disabled. All paths associated with these disabled design modes will be set to false path. No action is taken to any cell modes mapped to the disabled design modes.

Examples

In the following example, the first command defines two design modes, DM1 and DM2. (Since no mode group name was specified, the mode group is named "default".) The second command specifies that for design mode DM1, the READ cell mode is active for the RAM instance Uram1. The third command specifies that for design mode DM2, the WRITE component mode is active for the RAM instance Uram1. The fourth command maps all paths ending at Uram1/D0 to the design mode DM2.

```
pt_shell> define_design_mode_group {DM1 DM2}
pt_shell> map_design_mode DM1 READ Uram1
pt_shell> map_design_mode DM2 WRITE Uram1
pt_shell> map_design_mode DM2 -to Uram1/D0
```

The following example selects the design mode DM1 as the active mode for the current design, which in turn causes the cell mode READ to be selected as the active mode for the instance Uram1. Since design mode DM2 is automatically disabled all paths ending at Uram1/D0 become false paths.

```
pt_shell> set_mode -type design DM1
```

The following command directly selects the READ cell mode as the active mode for the RAM instance Uram1.

```
pt_shell> set_mode READ Uram1
```

The following command is equivalent to the previous command.

```
pt_shell> set_mode -type cell READ Uram1
```

See Also

- define_design_mode_group
- remove_design_mode
- reset_mode
- report_mode
- map_design_mode

Feedback

set_multi_input_switching_analysis

Sets inclusion or exclusion of multi-input switching (MIS) analysis for specified library cells or cell instances.

Syntax

int *set_multi_input_switching_analysis*

```
-exclude_only
   | -include_only
[-reset]
[-analysis_mode lib_cell | lib_arc | advanced]
object_list
```

Data Types

analysis_mode	one-of-string
object_list	list

Arguments

`-exclude_only`

Excludes the specified cells or library cells for MIS analysis.

`-include_only`

Includes only the specified cells or library cells for MIS analysis.

`-reset`

Resets a previously applied *set_multi_input_switching_analysis* specification.

`-analysis_mode lib_cell | lib_arc | advanced`

When used, indicates that the inclusion or exclusion list applies only to the specified type of MIS analysis. By default, the inclusion or exclusion list applies to all types of MIS analysis.

The *user_coefficient* value is deprecated; it is supported for compatibility as equivalent to *lib_cell*, but it will be obsoleted in a future release.

object_list

Specifies a list of cell instances or library cells for inclusion or exclusion of multi-input switching analysis.

Description

This command defines a list of cells or library cells to include or exclude in multi-input switching (MIS) analysis. It allows you to override the default behavior of which cells to analyze for MIS analysis.

Either the *-exclude_only* or *-include_only* option must be specified; they cannot both be specified.

Note the following:

- Specifying *-exclude_only* invalidates any previous *-include_only* for the same type of object specification (cell instances or library cells).
- Specifying *-include_only* invalidates any previous *-exclude_only* for the same type of object specification (cell instances or library cells).
- Specifying an *-include_only* list of library cells and an *-exclude_only* list of cell instances is allowed.
- Specifying an *-include_only* list of cell instances and an *-exclude_only* list of library cells is allowed.

Examples

The following example excludes MIS analysis for cell instance nand2_1:

```
pt_shell> set_multi_input_switching_analysis -exclude_only [get_cell
 nand2_1]
```

The following example excludes MIS analysis for library cell NAND2:

```
pt_shell> set_multi_input_switching_analysis -exclude_only [get_lib_cell
 NAND2]
```

The following example excludes all cells from MIS analysis, except for library cell NAND2:

```
pt_shell> set_multi_input_switching_analysis -include_only [get_lib_cell
 NAND2]
```

The following example excludes all cells from advanced MIS analysis except for library cell NAND2 (but still uses user-defined coefficients, if defined with the *set_multi_input_switching_coefficient* command):

```
pt_shell> set_multi_input_switching_analysis -analysis_mode advanced
 -include_only [get_lib_cell NAND2]
```

See Also

- si_enable_multi_input_switching_analysis
- si_multi_input_switching_analysis_mode

Feedback

set_multi_input_switching_coefficient

Sets delay coefficients for multi-input switching (MIS) analysis for a specified list of library cells.

Syntax

int *set_multi_input_switching_coefficient*

```
[-rise]
[-fall]
-delay coeff_value
object_list
```

Data Types

coeff_value	float
object_list	list

Arguments

`-rise`

Applies the coefficients to rise transitions only.

`-fall`

Applies the coefficients to fall transitions only.

`-delay` *coeff_value*

Indicates the base coefficient of the cell delay for multi-input switching analysis when the specified library cell is used.

object_list

Specifies a list of library cells for multi-input switching analysis.

Description

This command sets the base coefficients for the cell delay and delta delay of a victim net driven by this cell during multi-input switching (MIS) analysis. To enable multi-input switching analysis, set the *si_enable_multi_input_switching_analysis* variable to true; the default of this variable is false.

If the si_enable_multi_input_switching_timing_window_filter variable is set to true, the tool uses the base coefficient value to derive the actual MIS factor by considering the arrival windows of the input pins and their overlap analysis.

If the si_enable_multi_input_switching_timing_window_filter variable is set to false, the base coefficient value is directly applied to the MIS factor.

If you use the *set_multi_input_switching_coefficient* command multiple times on the same library cell, the last specified coefficient overwrites the previously specified coefficients.

Examples

The following example sets the delay coefficient value of 0.7 for all instances of library cell AND2 in the library MY_LIB:

```
pt_shell> set_multi_input_switching_coefficient -delay 0.7 [get_lib_cells
 MY_LIB/AND2]
```

See Also

- report_multi_input_switching_coefficient
- reset_multi_input_switching_coefficient
- si_enable_multi_input_switching_analysis
- si_enable_multi_input_switching_timing_window_filter

---

set_multicycle_path

Defines a multicycle path exception.

Syntax

status *set_multicycle_path*

```
[-setup]
[-hold]
[-rise]
[-fall]
[-start]
[-end]
[-reset_path]
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-comment comment_string]
[-sms_scenarios sms_scenarios_list]
path_multiplier
```

Feedback

Data Types

comment_string	string
fall_from_list	list
fall_through_list	list
fall_to_list	list
from_list	list
path_multiplier	integer
rise_from_list	list
rise_through_list	list
rise_to_list	list
sms_scenarios_list	collection
through_list	list
to_list	list

Arguments

`-setup`

Indicates that setup (maximum delay) calculations are to use the specified *path_multiplier*. Note that changing the *path_multiplier* for setup affects the default hold check as well. If neither *-setup* nor *-hold* is specified, *path_multiplier* is used for setup calculations and 0 is used for hold calculations.

`-hold`

Indicates that hold (minimum delay) calculations are to use the specified *path_multiplier*. Note that changing the *path_multiplier for setup affects the default hold check as well. If neither -setup nor -hold is specified, path_multiplier is used for setup calculations and 0 is used for hold calculations.*

`-rise`

Indicates that only rising path delays are to use *path_multiplier*. If neither *-rise* nor *-fall* is specified, both rising and falling delays are affected. "Rise" refers to a rising edge at the path endpoint.

`-fall`

Indicates that only falling path delays are to use *path_multiplier*. If neither *-rise* nor *-fall* is specified, both rising and falling delays are affected. "Fall" refers to a falling edge at the path endpoint.

`-start`

Indicates that the multicycle specification is relative to the period of the startpoint clock. The *-start* and *-end* options are needed only for multifrequency designs; otherwise, "start" and "end" specifications are equivalent.

The "start" clock is the clock source related to the register or primary input at the path startpoint. The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with *-start*

moves the relation backward one cycle of the start clock. A hold multiplier of 1 with *-start* moves the relation forward one cycle of the start clock.

`-end`

Indicates that the multicycle specification is relative to the period of the endpoint clock. The *-start* and *-end* options are needed only for multifrequency designs; otherwise, "start" and "end" specifications are equivalent.

The "end" clock is the clock source related to the register or primary output at the path endpoint. The default is to move the setup check relative to the end clock, and the hold check relative to the start clock. A setup multiplier of 2 with *-end* moves the relation forward one cycle of the end clock. A hold multiplier of 1 with *-end* moves the relation backward one cycle of the end clock.

`-reset_path`

Indicates that existing point-to-point exception are to be removed from the specified paths. If used with *-to* only, all paths leading to the specified endpoints are reset. If used with *-from* only, all paths leading from the specified startpoints are reset. If used with *-from* and *-to*, only paths between those points are reset. Only exceptions of the same rise/fall setup/hold type are reset. Using this option is equivalent to using the *reset_path* command with similar arguments before issuing the *set_multicycle_path* command.

`-from` *`from_list`*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified, one path startpoint on that cell is affected. You can use only one of the *-from*, *-rise_from*, and *-fall_from* options.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the *-from*, *-rise_from*, and *-fall_from* options.

`-fall_from` *`fall_from_list`*

Same as the *-from* option, except that the path must fall from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the *-from*, *-rise_from*, and *-fall_from* options.

`-through` *`through_list`*

Specifies a list of pins, ports, cells and nets through which the multicycle paths must pass. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you omit *-through*, all timing paths specified using the *-from* and *-to* options are affected.

`-rise_through` *`rise_through_list`*

This option is similar to the *-through* option, but applies only to paths with a rising transition at the specified objects.

`-fall_through` *`fall_through_list`*

This option is similar to the *-through* option, but applies only to paths with a fall transition at the specified objects.

`-to` *`to_list`*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected. You can use only one of the *-to*, *-rise_to*, and *-fall_to* options.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path. You can use only one of the *-to*, *-rise_to*, and *-fall_to* options.

`-fall_to` *`fall_to_list`*

Same as the *-to* option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path. You can use only one of the *-to*, *-rise_to*, and *-fall_to* options.

`-comment` *`comment_string`*

Associates a string description with the command for tracking purposes. When writing out SDC, this option is useful for tagging the methodology or tool that originally synthesized the command. You must begin and end the *comment_string* with double quotation marks (").

path_multiplier

An integer that specifies the number of cycles the data path must have for setup or hold, relative to the startpoint or endpoint clock, before data is required at the endpoint.

For example, specifying a *path_multiplier* of 2 for setup implies a 2-cycle data path. If you use *-setup*, *path_multiplier* is applied to setup path calculations. If you use *-hold*, *path_multiplier* is applied to hold path calculations. If you do not specify either *-setup* or *-hold*, *path_multiplier* is used for setup and 0 is used for hold. Note that changing the multiplier for setup affects the hold check as well.

`-sms_scenarios` *sms_scenarios_list*

Specifies a collection of SMS scenarios for which this multi cycle constraint is applied. When this option is not given the multi cycle constraint applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

This command specifies the number of clock cycles to adjust the setup or hold check of timing paths.

PrimeTime applies certain rules to determine single-cycle timing relationships for paths between clocked elements; the rules are based on active edges. For flip-flops, a single active edge both launches and captures data. For latches, the open edge launches data and the close edge latches data.

The setup check ensures that the correct data signal is available at destination registers in time to be correctly latched. The default setup behavior, called single-cycle setup, is as follows:

For multifrequency designs, there can be multiple setup relations between two clocks. For every latch edge of the destination clock, the setup check determines the nearest launch edge that precedes each latch edge. The smallest value of (setup_latch_edge - setup_launch_edge) determines the maximum delay requirement for this path.

You can override this default relationship by applying the *set_multicycle_path* or *set_max_delay* command to clocks, pins, ports, or cells. For example, setting the setup path multiplier to 2 with the *set_multicycle_path* command delays the latch edge one clock pulse. Note that changing the setup multiplier also affects the default hold check.

Most often, the setup check moves relative to the end clock. This move changes the data latch time at the path endpoint. In multifrequency designs, use the *set_multicycle_path -setup -start* command to move the data launch time backward. The hold check ensures that data from the source clock edge that follows the setup launch edge is not latched by the setup latch edge, and also ensures that data from the setup launch edge is not latched by the destination clock edge that precedes the setup latch edge.

The hold check is determined relative to each valid setup relationship, after applying multicycle path multipliers. The most restrictive (largest) difference of (hold_latch_edge - hold_launch_edge) is used as a minimum delay requirement for the path.

Important Note: The hold relationship is conservative. In some cases, you must override the default hold relationship. For multifrequency designs, it might might be more straightforward to use the *set_min_delay* command instead of the *set_multicycle_path -hold* command.

There are separate setup and hold multipliers for rise and fall. In most cases, these are set to the same value. You can use the *-rise* or *-fall* option to apply different values.

The *set_multicycle_path* command is a point-to-point timing exception command. The command can override the default single-cycle timing relationship for one or more timing paths. Other point-to-point timing exception commands include *set_max_delay*, *set_min_delay*, and *set_false_path*. False path exceptions always takes precedence over multicycle path exceptions. A more specific *set_max_delay* or *set_min_delay* command overrides a more general *set_multicycle_path* command.

When multiple exceptions commands of different types apply to the same path, they apply with the following precedence (highest to lowest):

```
1. set_false_path
2. set_max_delay and set_min_delay
3. set_multicycle_path
```

When multiple exception commands of the same type apply to a path, the more specific command overrides the more general. They apply with the following precedence (more specific to more general):

```
1. set_multicycle_path -from pin -to pin
2. set_multicycle_path -from pin -to clock
3. set_multicycle_path -from pin
4. set_multicycle_path -from clock -to pin
5. set_multicycle_path -to pin
6. set_multicycle_path -from clock -to clock
7. set_multicycle_path -from clock
8. set_multicycle_path -to clock
```

When multiple *set_multicycle_path* commands are applied that differ only by their cycle count, the last specification applies, even if it is less constraining than earlier specifications.

To undo a *set_multicycle_path* command, use the *reset_path* command. The *reset_design* command removes all attributes from the design, including specifications set by the *set_multicycle_path* command.

To disable setup and hold calculations for paths, use the *set_false_path* command. To list the point-to-point exceptions on a design, use the *report_exceptions* command.

When a path has transparent latches in the interior of the path, the *set_multicycle_path* command will only be applied if all the speficiers can be satisfied within one combinational segment between sequential elements. Commands using *-from* pin, and *-to* pin, with a transparent latch in between, will be ignored, because both specifiers could not be satisfied in a single combinational segment between sequential elements. When a command affects the Nth combinational segment, between the (N-1)th and Nth interior latches, the clock edge used by the Nth latch is adjusted according to the *path_multiplier*. All subsequent latches, and the final capture, are also adjusted. On such paths with interior transparent latches, it is possible for *set_multicycle_path* commands to affect each segement between sequential elements. The *path_multiplier* values for the various commands will affect corresponding segments. All the applied commands will affect the clock edge of the final capture on the path.

Examples

The following example sets all paths between latch1b and latch2d to 2-cycle paths for setup. Hold is measured at the previous edge of the clock at latch2d.

```
pt_shell> set_multicycle_path 2 -from { latch1b } -to { latch2d }
```

The following example moves the hold check to the preceding edge of the start clock.

```
pt_shell> set_multicycle_path -1 -from { latch1b } -to { latch2d }
```

The following example is a two-phase level-sensitive design, where the designer expects paths from phi1 to phi1 to be zero cycles.

```
pt_shell> set_multicycle_path 0 -from phi1 -to phi1
```

The following example uses *-start* to specify a 3-cycle path relative to the clock at the path startpoint. Clock sources are specified, affecting all sequential elements clocked by that clock, or ports with input or output delay relative to that clock.

```
pt_shell> set_multicycle_path 3 -start -from { clk50mhz } -to
 { clk10mhz }
```

The following example sets all rising paths to ff12/D to 2 cycles, and falling paths to 1 cycle.

```
pt_shell> set_multicycle_path 2 -rise -to { ff12/D }
pt_shell> set_multicycle_path 1 -fall -to { ff12/D }
```

The following multifrequency example shows a 20ns clock to 10ns clock with multicycle path of 2. Assume a path from ff1 (clocked by CLK2) to ff2 (clocked by CLK1).

```
pt_shell> create_clock -period 20 -waveform {0 10} CLK2
pt_shell> create_clock -period 10 -waveform {0 5} CLK1
pt_shell> set_multicycle_path 2 -setup -from ff1/CP -to ff2/D
```

The single-cycle setup relation is from the CLK1 edge at 0ns to the CLK2 edge at 10ns. With the multicycle path, the setup relation is now 20ns (from CLK1 edge at 0ns to CLK2 edge at 20ns). The hold relations are determined according to that setup relation.

1. Data from the source clock edge that follows the setup launch edge must not be latched by the setup latch edge. This implies a hold relation of 0ns (from CLK1 edge at 20ns to CLK2 edge at 20ns).
2. Data from the setup launch edge must not be latched by the destination clock edge that precedes the setup latch edge. This implies a hold relation of 10ns (from CLK1 edge at 0ns to CLK2 edge at 10ns).

The most restrictive (largest) hold relation is used, so the minimum delay requirement for this path is 10ns. This is a conservative check which might not apply to certain designs. Often you know that the destination register is disabled for the clock edge at 10ns. You can modify this default hold relation with the following to get an 0ns requirement.

```
pt_shell> set_multicycle_path 1 -hold -end -from ff1/CP -to ff2/D
```

However, a simpler approach is to use the following:

```
pt_shell> set_min_delay 0 -from ff1/CP -to ff2/D
```

The following example sets all timing paths from ff1/CP to ff2/D which passes through one or more of {U1/Z U2/Z} and one or more of {U3/Z U4/C} to 2 cycle paths for setup.

```
pt_shell> set_multicycle_path 2 -from ff1/CP -through {U1/Z U2/Z}
 -through {U3/Z U4/C} -to ff2/D
```

The following example applies both setup and hold multicycle path multipliers of 2 to two registers clocked by the same clock:

```
pt_shell> # don't do this - read explanation below
pt_shell> set_multicycle_path -setup -hold 2 -from ff1/CP -to ff2/D
```

This example results in incorrect (and optimistic) hold analysis. The meaning of the *-setup* and *-hold* multipliers are different - the *-setup* multiplier moves both the setup and hold checks forward, then the *-hold* multiplier moves the hold check backward from its setup-adjusted position. To ensure correct same-clock hold analysis, a setup multiplier of *N* should be accompanied by a hold multiplier of *N*-1:

```
pt_shell> set_multicycle_path -setup 2 -from ff1/CP -to ff2/D
pt_shell> set_multicycle_path -hold 1 -from ff1/CP -to ff2/D
```

For multifrequency designs, different multipliers or additional options (such as *-start* or *-end*) might be required.

Feedback

See Also

- current_design
- report_exceptions
- reset_design
- reset_path
- set_false_path
- set_max_delay
- set_min_delay

---

set_net_pattern

Set specific pattern for nets which will be used during multi-cycle vector generation or special mode vector generation.

Syntax

status *set_net_pattern*

```
[-object_list object_list]
[-file file_name]
[-strip_path strip_path]
[-format file_format]
[-start_time start_time]
[-mode one_hot | one_cold | constraint]
[-pattern pattern]
```

Data Types

```
object_list       list
file_name         string
strip_path        string
file_format       string
start_time        float
pattern           string
```

Arguments

`-object_list`

Specifies a list of net objects to be used with the option *-file*, *-pattern*, or *-mode*.

`-file`

Specifies a VCD file from which patterns will be read and applied on the list of nets in the option *-object_list*. The number of bits in the pattern is determined by number of cycles specifed in the option *set_vector_generation_options*

-number_of_cycles. Once the number of cycles is reached, the rest of the bits in the VCD file will be ignored. This option is only used with the options *-strip_path*, *-format*, or *-start_time*. This option is honored in multi-cycle generation only.

`-strip_path`

Specifies a path prefix that is to be stripped from all the object names read from the VCD file. This option is applied to strip the testbench and instance path from the VCD file.

`-format`

Specifies the format for the file in the options *-file*. VCD is the only format supported.

`-start_time`

Specifies start time from which patterns will be read from the VCD file. Patterns before this start time will be ignored.

`-mode`

Specifies mode used to generate a bit pattern for nets specified in the option *-object_list*. Accepted values are one_hot, one_cold, or constraint. Mode one_hot (or one_cold) is used to specify for a list of nets, where at any cycle, there should be one net stays at hot (or cold), and others stay at cold (or hot). When constraint mode is specified, a random bit pattern is generated based on the toggle rate and state probability from power analysis. This option is honored in multi-cycle vector generation only.

`-pattern`

Specifies a bit pattern where state value 0 or 1 is used to construct a bit string. The bit pattern is then applied cycle by cycle. This option is honored in either multi-cycle vector generation or special mode vector generation. Since special mode vector generation is for one clock cycle, only the first bit will be used in special mode.

Description

The *set_net_pattern* command specifies a bit pattern for a list of nets, which will be used during multi-cycle vector generation or special mode vector generation.

Examples

The following example reads patterns from a VCD file for all control nets, and sets one hot mode for a list of bus nets. For all primary input, constraint mode is applied. Then the command *write_vectors* is invoked to generate 100 cycle VCD.

```
pt_shell> set_vector_generation_options -number_of_cycles 100
pt_shell> set_net_pattern -file control.vcd -format VCD -strip_path tb
 -object_list [get_nets control*]
```

Feedback

```
pt_shell> set_net_pattern -mode one_hot -object_list [get_nets mbus[*]]
pt_shell> set_net_pattern -object_list [get_nets -of [get_ports]] -mode
 constraint
pt_shell> write_vectors 100_cycle.vcd
```

The following example shows how to generate a special mode VCD file, where all nets are considered as special nets, and 75% of those nets are toggling. At the same time, state 0 is set on all clock gating cell enable nets through the command *set_net_pattern*.

```
pt_shell> set_vector_generation_options -special_nets [get_nets -hier]
pt_shell> set_vector_generation_options -toggle_percentage 0.75
pt_shell> set_net_pattern -pattern 0 -object_list [get_nets -of [get_pins
 icg_*/enble]
pt_shell> write_vectors special_mode_75.vcd
```

See Also

- set_vector_generation_options
- write_vectors

---

set_noise_derate

Sets noise derating information for the current design. This does not affect aggressor screening and filtering result.

Syntax

int *set_noise_derate*

```
[-above]
[-below]
[-low]
[-high]
[-height_offset hoffset]
[-absolute_height_offset_in_volts abshoffset]
[-height_factor hfactor]
[-width_factor wfactor]
[object_list]
```

Data Types

hoffset	float
abshoffset	float
hfactor	float
wfactor	float
object_list	list

Feedback

Arguments

`-above`

Specifies a derating for above ground or power rails noise analysis region.

`-below`

Specifies a derating for below ground or power rails noise analysis region.

`-low`

Specifies a derating for ground rail noise.

`-high`

Specifies a derating for power rail noise.

`-height_offset` *`hoffset`*

Specifies an offset voltage in ratio of Vdd, that is directly added to each aggressor and total noise height of a pin in the design. The value of this offset must be between -1.0 to 1.0. The default for this offset is 0.0.

`-absolute_height_offset_in_volts` *`abshoffset`*

Specifies an offset voltage in absolute voltage units that is directly added to each aggressor and total noise height of a pin in the design. The default for this offset is 0.0.

`-height_factor` *`hfactor`*

Specifies a scale factor that could increase or decrease the calculated height of the noise waveform of each aggressor and the total noise height of a pin in the design. The value for this scale factor must be larger than 0.0. The default for this scale factor is 1.0.

`-width_factor` *`wfactor`*

Specifies a scale factor that could increase or decrease the calculated width of the noise waveform of each aggressor and the total noise width of a pin in the design. The value for this scale factor must be larger than 0.0. The default for this scale factor is 1.0.

`object_list`

Specifies a list of input pins, output ports or library pins

Description

This command sets the noise derate offset and scale factors

- On the current design if you do not use the *object_list* option
- On the specified pins and ports if you use the *object_list* option

Feedback

Noise derating offset and scale factors

- Affect the noise height and width values shown in noise reports
- Affect the noise propagated through stages
- Do not affect aggressor filtering or screening

If the offset is not specified for a noise analysis region, 0.0 offset is assumed. The absolute and relative height offsets can be specified together. If scale factors are not specified for a noise analysis region, 1.0 scale factors are assumed.

To show derating offsets and scale factors settings, use the *report_noise_calculation* command.

Examples

This example specifies a voltage offset of 0.10 and noise height and width scale factors of 0.90 for above the ground rail noise:

```
pt_shell> set_noise_derate -above -low -height_offset 0.10 \\
          -height_factor 0.90 -width_factor 0.90
```

To restore the settings to their original values:

```
pt_shell> set_noise_derate -above -low -height_offset 0.0 \\
          -height_factor 1.0 -width_factor 1.0
```

See Also

- report_noise
- report_noise_calculation
- set_noise_parameters

---

set_noise_immunity_curve

Specifies the noise immunity curve at an input of a library cell or at an output port of the design.

Syntax

status *set_noise_immunity_curve*

```
[-above]
[-below]
[-low]
[-high]
[-height height_value]
[-width  width_value]
```

```
[-area   area_value]
object_list
```

Data Types

```
height_value     float
width_value      float
area_value       float
object_list      list
```

Arguments

`-above`

Specifies an immunity curve for above ground or power rail noise analysis region.

`-below`

Specifies an immunity curve for below ground or power rail noise analysis region.

`-low`

Specifies an immunity curve for ground rail noise. When used with the *-above* or *-below* option, the *-low* option specifies the immunity curve for above-low or below-low rail noise, respectively.

`-high`

Specifies an immunity curve for power rail noise. When used with the *-above* or *-below* option, the *-high* option specifies the immunity curve for above-high or below-high rail noise, respectively.

`-height` *`height_value`*

Specifies the height offset of the noise immunity curve above the horizontal axis, in the voltage units of the library. A smaller value increases the likelihood that violations are reported.

`-width` *`width_value`*

Specifies the width offset of the noise immunity curve from the vertical axis, in time units of the library. A smaller value increases the likelihood that violations are reported.

`-area` *`area_value`*

Specifies a relative area under the hyperbolic curve, in units equal to the product of the voltage units and time units of the library.

`object_list`

Specifies a list of library pins or ports.

Feedback

Description

Each cell input can tolerate a certain amount of noise without causing a failure at the cell output. This characteristic is called noise immunity. This command specifies noise immunity at library input pins or design output ports to determine whether noise failures occur as well as the amount of noise slack.

Noise immunity in the library can be specified as noise immunity curves, polynomials, tables, or in terms of noise margins that consider only the bump heights at the cell inputs.

This command specifies a noise immunity curve at an input of a library cell or at an output port of the design. Use this command in the absence of library-specified noise immunity characteristics, or to override the library-specified characteristics by replacing them with a noise immunity curve. This command also overrides noise margins that have been annotated using the *set_noise_margin* command,

The noise immunity curve is a hyperbola described by the following equation:

y = C1 + C2 / (x - C3)

where

- y = noise immunity height in library voltage units
- x = bump width in library time units
- C1 = height offset set with the *-height* option
- C2 = area parameter set with the *-area* option
- C3 = width offset set with the *-width* option

A noise violation is reported when bump height is above the hyperbola, or in other words,

bump_height > C1 + C2 / (bump_width - C3)

Specify the three constants C1, C2, and C3 (the *-height*, *-area*, and *-width* options) so that the hyperbolic curve matches the noise immunity points obtained by circuit simulation of the cell containing the pin.

Noise immunity characteristics can vary for different noise bump types, so you can specify four different noise immunity curves associated with each input: below low, above low, below high, and above high. Specify the C1, C2, and C3 constants as positive numbers for all four types of noise bumps.

The *report_noise_calculation* command shows whether noise immunity information was taken from the library or annotated by this command.

Feedback

Examples

This example specifies a noise height offset of 0.59 voltage units, a noise width offset of 0.04 time units, and a hyperbolic area constant of 0.0064 voltage-time units for above-the-ground rail noise at the A pin of the IV library cell in the lsi_10k library:

```
pt_shell> set_noise_immunity_curve -above -low -height 0.59 -width 0.04
 \\
            -area 0.0064 lsi_10k/IV/A
```

A violation is reported when bump_height > 0.59 + 0.0064 / (bump_width - 0.04).

See Also

- remove_noise_immunity_curve
- report_noise_calculation
- set_noise_margin

---

set_noise_lib_pin

Sets an equivalent noise library pin for a driver or load.

Syntax

int *set_noise_lib_pin*

```
pins
lib_pin
```

Data Types

```
pins        list
lib_pin     list
```

Arguments

pins

Specifies a collection of pins for which the noise library pin is set.

lib_pin

Specifies the equivalent library pin from which the noise information is used.

Description

This command allows to redefine the noise information for a certain library pin. For output pins, setting an equivalent noise library pin means that the I/V curves of the equivalent noise library pin is used. For input pins, noise immunity information of the equivalent noise library pin is used.

Feedback

Examples

This example specifies an equivalence for noise information on two inputs pins of the design to be the same as an input library pin:

```
pt_shell> set_noise_lib_pin [get_pins {cell/pin1 cell/pin2}]
 lsi_10k/IV/A\\
```

See Also

- update_noise
- report_noise
- report_noise_calculation

---

set_noise_margin

Specifies the noise margin for a library pin, port, or pin.

Syntax

status *set_noise_margin*

```
[-above]
[-below]
[-low]
[-high]
margin_value
object_list
```

Data Types

margin_value	float
object_list	list

Arguments

`-above`

Specifies the noise margin for above ground or power rail noise analysis region.

`-below`

Specifies the noise margin for below ground or power rail noise analysis region.

`-low`

Specifies the noise margin for ground rail noise.

`-high`

Specifies the noise margin for power rail noise.

margin_value

Specifies a margin value. The value is the input height in voltage units.

object_list

Specifies a list of library pins, ports, or pins.

Description

Each library cell input can tolerate a certain amount of noise without causing a failure at the cell output. This characteristic is called noise immunity. This command specifies noise immunity at library pins to determine whether noise failures occur as well as the amount of noise slack.

Noise immunity in the library can be specified as noise immunity curves, polynomials, or tables or in terms of noise margins.

This command specifies a noise margin for a library pin, design port, or pin. It can be used in the absence of library-specified noise immunity characteristics, or to override the library-specified characteristics by replacing them with noise margins. Noise immunity curves that have been annotated using the *set_noise_immunity_curve* command overrides noise margins set by this command.

Noise margins consider only the bump heights at the cell inputs. Using height-only noise margins are faster and more conservative than the other methods.

For high, narrow noise bumps, using height-only noise margins is pessimistic because it treats some bumps as noise failures that would otherwise pass with the immunity curve model. However, for wide noise bumps, using noise margins gives the same results as using noise immunity curves.

Noise immunity characteristics can vary for different noise bump types, so there can be four different noise margins associated with each input: below low, above low, below high, and above high. Specify all values as positive numbers for all four types of noise bumps.

In the absence of command-specified or library-specified noise immunity data, the tool calculates the maximum allowable noise bump heights based on DC noise margins of the driver and receiver, as defined in the .lib logic library by the input/output logic-level parameters.

The *report_noise_calculation* command shows whether noise margin information was taken from the library or annotated by this command.

Examples

This example specifies a noise margin of 4 for above-the-ground rail noise for the A pin of the IV library cell in the lsi_10k library:

```
pt_shell> set_noise_margin -above -low 4 lsi_10k/IV/A
```

Feedback

See Also

- remove_noise_margin
- report_noise_calculation
- set_noise_immunity_curve

---

set_noise_parameters

Defines the noise analysis parameters for the current design.

Syntax

status *set_noise_parameters*

```
[-ignore_arrival]
[-include_beyond_rails]
[-enable_propagation]
[-analysis_mode report_at_source | report_at_endpoint]
[-analysis_type violators | all]
```

Arguments

`-ignore_arrival`

Ignores the arrival window information of the aggressors during noise analysis. Therefore, the aggressors are assumed to be always overlapping to maximize the effect of a coupled noise bump.

`-include_beyond_rails`

Enables the analysis of noise beyond the high and low regions. If you do not use this option, the analysis of noise above the high rail and below the low rail is disabled.

`-enable_propagation`

Enables noise propagation. Propagated noise on a victim net is caused by noise at an input of the cell that is driving the victim net. PrimeTime SI can calculate propagated noise at a cell output, given the propagation characteristics of the cell, the noise bump at the cell input, and the load on the cell output.

`-analysis_mode report_at_source | report_at_endpoint`

Specifies one of the following analysis modes:

- *report_at_source* (the default) - Reports violations at the source of violations.
- *report_at_endpoint mode* - Propagates violations through fanout and reports violations at endpoints.

Feedback

`-analysis_type violators | all`

Focuses the noise analysis with one of the following analysis types; both analysis types catch all noise violations:

- *violators* (the default) - Evaluates each net in the design for potential noise violations. The tool performs detailed noise analysis on only nets with potential violations. This analysis type is particularly useful if the main purpose of noise analysis is to detect noise violations. This mode can be considerably faster than the *all* analysis type.
- *all* - Performs detailed noise analysis on all nets, regardless of whether they have potential violations.

Description

This command specifies the parameters that are considered during the noise analysis. If you do not use this command, or you use the *reset_noise_parameters* command, the tool follows the default noise analysis behavior -- the tool ignores beyond-the-rail analysis, considers the arrival times of aggressors, and ignores propagated noise.

If you do not specify any options with this command, the command has no effect, and the tool follows the default noise analysis behavior.

To report the status of the noise analysis parameters for the current design, use the *report_noise_parameters* command.

To reset the noise analysis parameters to the default settings, use the *reset_noise_parameters* command.

Examples

In the following example, the noise propagation is enabled while other parameters are not affected.

```
pt_shell> set_noise_parameters -enable_propagation
```

See Also

- get_noise_violation_sources
- report_noise
- report_noise_parameters
- report_noise_violation_sources
- reset_noise_parameters
- update_noise

Feedback

set_ocvm_table_group

Associates an AOCV or POCV named table group or Liberty-based AOCV or POCV derating group with the list of specified design objects.

Syntax

status *set_ocvm_table_group*

```
ocvm_table_group_name
-type aocvm | pocvm
[-table_type coefficient | distance]
[object_list]
```

Data Types

```
ocvm_table_group_name     string
object_list               list
```

Arguments

ocvm_table_group_name

Specifies the name of the AOCV or POCV table group.

`-type aocvm | pocvm`

Specifies AOCV or POCV tables.

`-table_type coefficient | distance`

Specifies coefficient or distance-based POCV tables. The default is *coefficient*. Do not use this option for AOCV.

object_list

Specifies the list of design objects.

Description

The *set_ocvm_table_group* command can operate on either hierarchical cells / top level design or on lib_cells.

When operating on hierarchical cells or a top-level design, the command sets the association between a named AOCV or POCV table group and a hierarchical instance. An AOCV or POCV table group is a collection of AOCV or POCV tables with the same name. Names are specified using the group_name field in the AOCV or POCV table syntax. When associations are specified between hierarchical cells in the design and named AOCV or POCV table groups, a cell (or net) derives its AOCV or POCV derating from the table group associated with the hierarchical cell (or design) that (a) fully contains the cell (or net), and (b) is the lowest-level (closest ancestor) hierarchical cell (or design) with an associated AOCV or POCV table group. If no such hierarchical cell (or design) containing

the cell (or net) is found, the unnamed AOCV or POCV table group is used. Note that a net inherits the AOCV or POCV derating set of a hierarchical cell (design) that fully contains the net. For example, for nets that cross the boundary between the top level and a block (with an associated AOCV or POCV table group), the lowest-level hierarchical cell that fully contains the net is not the block. Therefore, the net derating used comes from an AOCV or POCV table group that contains the block and fully encloses the net. Note that for POCV, table groups for coefficients and distance-based tables are independent.

When operating on lib_cells the *set_ocvm_table_group* command can be used to assign a Liberty based AOCVM table from a Liberty library to the given lib cell. The Liberty AOCVM format allows for tables, which are not assigned to any library cell by default, to still be referenced by name. This command provides the ability to setup this reference. The syntax for the P/AOCVM derate group to be referenced follows this pattern:

lib_name/[*lib_cell_name*]/*derate_group_name*

The *lib_cell_name* is optional and applicable only for lib_cell based table. A lib_cell based can only be assigned to a library cell of the same name.

Examples

In the following example, the *set_ocvm_table_group* command performs an association between the hierarchical cell H1 and the named POCV coefficient table group A1. A POCV coefficient table belonging to the A1 table group is also shown.

```
pt_shell> set_ocvm_table_group -type pocvm -table_type coefficient A1
 [get_cells H1]

pt_shell> sh cat test.pocvm

version:        4.0
ocvm_type:      pocvm
group_name:     A1
object_type:    lib_cell
rf_type:        rise
delay_type:     cell
derate_type:    late
object_spec:    my_lib/*
coefficient:    0.05
```

See Also

- get_timing_paths
- report_ocvm
- report_timing
- reset_ocvm_table_group

Feedback

set_operating_conditions

Defines the operating conditions or environmental characteristics for the current design.

Syntax

status *set_operating_conditions*

```
[-analysis_type single | on_chip_variation]
[-library lib]
[condition]
[-min min_condition]
[-max max_condition]
[-min_library min_lib]
[-max_library max_lib]
[-object_list objects]
```

Data Types

```
lib                 string
condition           string
min_condition       string
max_condition       string
min_lib             string
max_lib             string
objects             list
```

Arguments

`-analysis_type single | on_chip_variation`

Specifies how the specified operating conditions are used:

- *single* - Specifies that only one operating condition is to be used.
- *on_chip_variation* - Switches the design to min_max mode; specifies that the minimum and maximum operating conditions represent, respectively, the lower and upper bounds of the maximum variation of operating conditions on the chip. All maximum path delays use the maximum operating condition, and all minimum path delays use the minimum operating condition.

`-library lib`

Specifies the library that contains definitions of the environmental characteristics to be used. This is either a library name or a collection object.

`condition`

Specifies the name of the single operating condition.

`-min` *`min_condition`*

Specifies the minimum operating condition used for checking minimum delays and hold violations (see the *report_timing* command with the *-delay min* option). If you use this option, you must also use the *-max* option.

`-max` *`max_condition`*

Specifies the maximum operating condition used for checking maximum delays and setup violations (see the *report_timing* command with the *-delay max* option). If you use this option, you must also use the *-min* option.

`-min_library` *`min_lib`*

Specifies the library that contains the *min_condition*. This is either a library name or a collection object. If you use this option, you must also use the *-min* option.

`-max_library` *`max_lib`*

Specifies the library that contains the *max_condition*. This is either a library name or a collection object. If you use this option, you must also use the *-max* option.

`-object_list` *`objects`*

Applies the operating conditions on the specified the cells or ports. If you do not use this option, the operating conditions are applied to the entire design. This option accepts both leaf cells and hierarchical blocks. You cannot use this option with the *-analysis_type* option. This option is only supported in legacy flows; use the *set_voltage* command instead.

Description

This command defines the operating conditions (or environmental characteristics) for performing timing analysis on the current design. You are in min_max mode if you use the *-min* and *-max* options.

The specified operating conditions must be defined in one of the following ways:

- Use the *min_lib* value for only the *min_condition*
- Use the *max_lib* value for only the *max_condition*
- Use one of the libraries in the *link_path*

The order for a library search is as follows:

•

1. *lib* • 2. *min_lib* or *max_lib* • 3. *link_path*

If no operating conditions are specified for a design, the tool uses the default operating condition of the library to which the cell is linked. If the library does not have a default operating conditions, no operating conditions are used.

Operating conditions set by using the *-object_list* option override the operating conditions set on design or higher levels of hierarchy.

To view the operating conditions defined for the current design and to determine the libraries to which the current design are linked, use the *report_design* command.

To view the operating conditions defined in the specified library, use the *report_lib* command.

To view the nondefault operating conditions set on cells, use the *report_cell* command.

To remove operating conditions from the current design, use the *remove_operating_conditions* or *reset_design* command.

When process factor, operating temperature, and operating voltage deviate from their nominal values, the tool uses delay scaling among multiple libraries.

PrimeTime derives cell instance voltages as follows:

- Default supply voltage in library cell definition (voltage_map in Liberty syntax)
- *set_operating_conditions* applied at the design level
- *set_voltage* on a supply net
- *set_voltage* on a PG pin

Since operating condition defines only single voltage value caution should be exercised when setting operating conditions on designs containing multirail cells such as level shifters and power management cells. PrimeTime applies the operating condition voltage value to all the library power rail.

The correct way to completely avoid this issue on multirail cells is to use the proper multivoltage flow, such as the UPF flow, to define all design rails using the *set_voltage* command. To verify voltage values of all rails of a multirail cell use the *report_power_pin_info* command.

An alternative is not to use any *set_operating_conditions* command and thus all power rails default to the library nominal voltage defined by voltage_map. Otherwise, use the design-wide *set_operating_conditions*; however, add the *set_voltage* command on specific PG pins of all multirail cells. Also, Liberty libraries should be created with the first voltage_map to match the nominal voltage nom_voltage.

Note that if the analysis type is set to *single*, only maximum delays from a SDF file are used to annotate timing arcs. Changing from a different analysis type into the single

operating condition analysis causes all minimum delays to be overwritten by maximum delay values.

Examples

The following example shows an operating condition definition, as it appears in the source text of a library.

```
operating_conditions("BCCOM") {
  process : 0.6 ;
  temperature : 20 ;
  voltage : 5.25 ;
  tree_type : "best_case_tree" ;
}
```

The name of this set of operating conditions is BCCOM. The parameters are defined as follows:

- *process* - A floating-point number that represents the characteristics of a semiconductor manufacturing process.
- *temperature* - A floating-point number that represents the temperature of the defined environment.
- *voltage* - A floating-point number that defines the upper boundary of the voltage range in which the defined environment operates. The lower boundary is always 0.0.
- *tree-type* - The interconnect model for the environment. The tool uses the interconnect model to select a formula for calculating interconnect delays. Three models are available:
  - *best_case_tree* - Uses a lumped RC model.
  - *worst_case_tree* - All loads assume full wire resistance.
  - *balanced_tree* - All loads share the wire resistance evenly.

The following example uses operating condition WCIND found in the my_lib.db library to set the operating conditions to WCIND if the *link_path* is my_lib.db.

```
pt_shell> set_operating_conditions WCIND
```

The following example uses the WCIND operating conditions in the other_lib.db library:

```
pt_shell> set_operating_conditions WCIND -library other_lib.db
```

In the following example, the tool uses

Feedback

- BCCOM values for the minimum operating conditions when calculating minimum delays
- WCCOM values for the maximum operating conditions when calculating maximum delays

```
pt_shell> set_operating_conditions -min BCCOM -max WCCOM \\
          -analysis_type on_chip_variation
```

See Also

- create_operating_conditions
- define_scaling_lib_group
- remove_operating_conditions
- report_cell
- report_design
- report_lib
- report_power_pin_info
- report_timing
- reset_design
- set_temperature
- set_voltage
- default_oc_per_lib

---

set_opposite

Sets two ports to be logically opposite.

Syntax

```
status set_opposite
  port1
  port2
```

Data Types

```
port1      string
port2      string
```

Feedback

Arguments

port1

Specifies the first input port.

port2

Specifies the second input port.

Description

This command defines two input ports in the current design as logically opposite. This information is used during synthesis to eliminate redundant ports to improve optimization quality.

Logical inconsistencies are checked in relationships between ports. Specifying an inconsistent logical relationship between ports is not allowed.

Use the *reset_design* command to remove this property from a port.

The logical relationship is characterized by the *characterize_context* command and exported to synthesis using the *write_context* command.

Examples

The following example sets two input ports "A" and "B" to be logically opposite.

```
pt_shell> set_opposite A [get_ports B]
1
```

The following example sets up a contradictory relationship between two ports and generates an error message.

```
pt_shell> set_opposite A B
1

pt_shell> set_equal A B
Error: Can't set opposite ports equal in design 'top': 'A' 'B'. (DDB-23)
0
```

See Also

- set_equal
- characterize_context
- write_context

---

set_output_delay

Sets output path delay values for the current design.

Feedback

Syntax

string *set_output_delay*

```
[-clock clock_name]
[-reference_pin pin_port_name]
[-clock_fall]
[-level_sensitive]
[-rise]
[-fall]
[-max]
[-min]
[-add_delay]
[-network_latency_included]
[-source_latency_included]
[-group_path group_name]
[-sms_scenarios sms_scenarios_list]
delay_value
port_pin_list
```

Data Types

clock_name	list
delay_value	float
group_name	string
pin_port_name	list
port_pin_list	list
sms_scenarios_list	collection

Arguments

`-clock` *clock_name*

Specifies the name of a clock to which the specified delay is to be related. If you specify the *-clock_fall* option, you must also specify the *-clock* option.

`-reference_pin` *pin_port_name*

Specifies the clock pin or port to which the specified delay is related. If you use this option, and if propagated clocking is used, the delay value is related to the arrival time at the specified reference pin, which is clock source latency plus its network latency from the clock source to this reference pin. The *-network_latency_included* and *-source_latency_included* options cannot be used at the same time as the *-reference_pin* option. For ideal clock network, only source latency is applied.

The pin specified with the *-reference_pin* option should be a leaf pin or port in a clock network, and in the direct or transitive fanout of a clock source specified with the *-clock* option. If multiple clocks reach the port or pin where you are setting the input delay, and if the *-clock* option is not used, the command considers all of the clocks.

`-clock_fall`

Indicates that the delay is relative to the falling edge of the clock. If you specify the *-clock_fall* with the *-reference_pin* option, the delay is relative to the falling transition of the reference pin. If you specify the *-clock* option, the default is the rising edge or the rising transition of the reference pin. If you specify the *-clock_fall* option, you must also specify the *-clock* option.

`-level_sensitive`

Indicates that the destination of the delay is a level-sensitive latch. This allows the tool to derive a setup and hold a relationship for paths to this port as if it were a level-sensitive latch. If you do not use the *-level_sensitive* option, the output delay is treated as if it were a path to a flip-flop.

`-rise`

Indicates that the *delay_value* option refers to a rising transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* options, rising and falling delays are assumed to be equal.

`-fall`

Indicates that the *delay_value* option refers to a falling transition on specified ports of the current design. If you do not specify the *-rise* or *-fall* options, rising and falling delays are assumed to be equal.

`-max`

Indicates that the *delay_value* option refers to the longest path. If you do not specify the *-max* or *-min* options, maximum and minimum output delays are assumed to be equal.

`-min`

Indicates that the *delay_value* option refers to the shortest path. If you do not specify the *-max* or *-min* options, maximum and minimum output delays are assumed to be equal.

`-add_delay`

Indicates that delay information is added to the existing output delay, instead of overwriting the output delay. By using the *-add_delay* option, you can capture information about multiple paths leading from an output port that are relative to different clocks or clock edges. For example, the *set_output_delay 5.0 -max -rise -clock phi1 {OUT1}* command removes all other maximum rise output delays from *OUT1*, because the *-add_delay* option is not specified. Other output delays with a different clock or with the *-clock_fall* option are removed.

In another example, the *-add_delay* option is specified: *set_output_delay 5.0 -max -rise -clock phi1 -add_delay {Z}*. If there is an output maximum rise delay for Z relative to the clock phi1 rising edge, the larger value is used. The smaller

value does not result in critical timing for maximum delay. For minimum delay, the smaller value is used. If there is a maximum rise output delay relative to a different clock or different edge of the same clock, it remains with the new delay.

`-network_latency_included`

Specifies whether the clock network latency should not be added to the output delay value. If this option is not specified, the clock network latency of the related clock is added to the output delay value. It has no effect if the clock is propagated or the output delay is not specified with respect to any clock.

`-source_latency_included`

Specifies whether the clock source latency should not be added to the output delay value. If this option is not specified, the clock source latency of the related clock is added to the output delay value. It has no effect if the output delay is not specified with respect to any clock.

`-group_path` *`group_name`*

Specifies the name of a group into which paths ending at the specified ports or pins are added. If the group does not already exist, one is created. The *-group_path* option of the *set_output_delay* command is equivalent to specifying the *group_path -name group_name -to port_pin_list* command. If you do not specify the *-group_path* option of the *set_output_delay* command, the existing path grouping does not change.

`delay_value`

Specifies the path delay in units consistent with the technology library used during analysis. The *delay_value* option represents the amount of time before a clock edge that the signal is required. For maximum output delay, this represents a combinational path delay to a register plus the library setup time of that register. For minimum output delay, this value is usually the shortest path delay to a register minus the library hold time.

`port_pin_list`

Specifies a list of output port or internal pin names in the current design to which the *delay_value* option is assigned. If more than one object is specified, the objects are enclosed in braces ({}).

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this output delay is applied. When this option is not given the output delay applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Feedback

Description

This command sets output path delay values for the current design. The input and output delays characterize the operating environment of the current design when used with the *set_load* and *set_driving_cell* commands.

The *set_output_delay* command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless specified. For inout (bidirectional) ports, you can specify the path delays for both input and output modes.

To describe a path delay to a level-sensitive latch, use the *-level_sensitive* option. If the latch is positive-enabled, set the output delay relative to the rising clock edge; if it is negative-enabled, set the output delay relative to the falling clock edge. If time is borrowed at that latch, subtract that time from the path delay to the latch when determining output delay.

The *characterize_context* command automatically sets input and output delay, drive, and load values based on the environment of a cell instance.

PrimeTime adds input delay to path delay for paths starting at primary inputs and to output delay for paths ending at primary outputs.

The *-group_path* option modifies the path grouping. Path grouping affects the maximum delay cost. The worst violator within each group adds to the cost.

If a reference pin is not reachable by any given clock, zero arrival time is assumed. If no clock is specified with a reference pin and this reference pin is not reachable by any clock, an unconstrained path is reported. This behavior is changed after X0506. The new behavior of these invalid constraints are ignored and path is constrained by whatever it should be, as if no such constraints were defined. The *check_timing* command generates a warning if the reference pin is not reachable by any active clock, or if multiple clocks reach the reference pin and no clock is specified in the command.

To get a report on the latency calculation using a reference pin, use the *report_timing -path_type full_clock* or *report_timing -path_type full_clock_expanded* commands.

To list output delays associated with ports, use the *report_port* command. To list output delays of internal pins, use the *report_timing* command. To list the path groups that are defined, use the *report_path_group* command.

To remove output delay values, use the *remove_output_delay* or *reset_design* commands. To modify paths grouped with the *-group_path* option, use the *group_path* command to place the paths in another group or the default group.

Examples

The following example sets an output delay of 1.7 relative to the rising edge of CLK1 for all output ports in the design.

```
pt_shell> set_output_delay 1.7 -clock CLK1 [all_outputs]
```

The following example sets the input and output delays for the bidirectional port INOUT1. The input signal arrives at INOUT1 2.5 units after the falling edge of CLK1. The output signal is required at INOUT1 at 1.4 units before the rising edge of CLK2.

```
pt_shell> set_input_delay 2.5 -clock CLK1 -clock_fall {INOUT1}

pt_shell> set_output_delay 1.4 -clock CLK2 {INOUT1}
```

The following example models the situation where there are three paths from output port OUT1. One of the paths is relative to the rising edge of CLK1. Another path is relative to the falling edge of CLK1. The third path is relative to the falling edge of CLK2. The *-add_delay* option indicates that new output delay information does not cause the removal of old information.

```
pt_shell> set_output_delay 2.2 -max clock CLK1 -add_delay {OUT1}

pt_shell> set_output_delay 1.7 -max clock CLK1 -clock_fall -add_delay
 {OUT1}

pt_shell> set_output_delay 4.3 -max clock CLK2 -clock_fall -add_delay
 {OUT1}
```

In the following example, two different maximum delays and two minimum delays for port Z are specified using the *-add_delay* option. Because the information is relative to the same clock and clock edge, only the largest of the maximum values and the smallest of the minimum values are maintained (in this case, 5.0 and 1.1). If the *-add_delay* option is not used, the new information overwrites the old information.

```
pt_shell> set_output_delay 3.4 -max -clock CLK1 -add_delay {Z}

pt_shell> set_output_delay 5.0 -max -clock CLK1 -add_delay {Z}

pt_shell> set_output_delay 1.1 -min -clock CLK1 -add_delay {Z}

pt_shell> set_output_delay 1.3 -min -clock CLK1 -add_delay {Z}
```

The following example shows how to use the *-group_path* option to add ports into a named group. Note that without this option, paths to these ports are included in the CLK group.

```
pt_shell> set_output_delay 4.5 -max -clock CLK -group_path busA {busA[*]}
```

See Also

- all_outputs
- characterize_context
- create_clock

- current_design
- group_path
- remove_output_delay
- report_design
- report_path_group
- report_port
- reset_design
- report_timing
- check_timing
- set_driving_cell
- set_load
- set_max_delay
- set_output_delay

---

set_parasitic_corner

Sets a parasitic corner for the timing analysis in the presence of variation-aware parasitics.

Syntax

status *set_parasitic_corner*

```
-name corner_name
file_name
```

DATA TYPES

file_name string *corner_name* string

Arguments

`-name corner_name`

Specifies the name of the parasitic corner to be used from the corner file. A given corner file can contain definitions of a number of parasitic corners and this option specifies which corner definition is to be used to be set as the corner for analysis.

file_name

Specifies the name of the parasitic corner or corner file. The file is supposed to contain the percentage variation values for all the parasitic parameters in the variation-aware parasitics.

Description

The *set_parasitic_corner* command reads parasitic corner information from a disk file and sets it as the corner for analysis in the presence of variation-aware parasitics. The command succeeds only if variation-aware parasitics were previously annotated.

The syntax of the corner parasitic file is very simple. It simply defines various parasitic corners by specifying the VARIATION_RATIOs for all the parameters.

A VARIATION_RATIO signifies the actual amount of variation from the mean (or nominal) for a given parameter. A VARIATION_RATIO is expressed in terms of the percentage variation of a given parameter with respect to its variation coefficient. For example, a given parameter has a variation coefficient of 0.09. The actual amount of variation allowed for this parameter is between -0.27 and 0.27. The VARIATION_RATIO is the multiplier to the variation coefficient that gets us to the actual amount of variation for the parameter. So, by definition, the value of VARIATION_RATIO has to be between -3.0 and +3.0.

Technically, the syntax of a corner file can be expressed as:

```
(CORNER_NAME: <name of the corner>  (PARAM_NAME PARAM_VARIATION_RATIO)*)+
```

The corner file should contain definition for at least one corner. A corner can contain VARIATION_RATIO values for ZERO or more number of parameters. If a parameter is not specified in the corner definition, the VARIATION_RATIO is assumed to be ZERO.

Comments can be written in the file following Tcl style - so, anything after the characters " #" (space and pound) on a line is assumed to be a comment until the end of the line. Only line comments are supported.

An example of the corner file is as follows:

```
CORNER_NAME: CMAX
     M1_W         3.0
     M2_T         3.0
     ILD_c_T     -0.04
CORNER_NAME: RCMAX
 ## my definition of RCMAX
     M1_W         3.0
     M2_T         2.0
     ILD_c_T     -0.22
CORNER_NAME: RCMIN
     M1_W         -3.0
     M2_T         -3.0
     ILD_c_T       0.4
```

Feedback

This indicates that the CMAX corner parasitics must be inferred where the M1_W parameter is changed by positive (3.0 * variation_coeff of M1_W) from the nominal and so on.

You can issue this command multiple times to change the corner for analysis. Only the last command takes effect. You can remove the parasitic corner setting by using the *remove_parasitic_corner* command.

Note that this command has no effect on the *write_parasitics* command. This command sets the corner only for analysis and does not change the variation-aware parasitics.

The parasitic attributes, such as the *rc_network* and *total_coupling_capacitance* attributes, are all modified to reflect the corner values.

Examples

The following example reads the parasitics corner file "./corner_file" from disk and uses the settings of the RCMAX corner to set the default parasitic corner on the design with variation-aware parasitics.

```
pt_shell> set_parasitic_corner -name RCMAX ./corner_file
```

See Also

- remove_parasitic_corner
- read_parasitics
- report_annotated_parasitics

---

set_parasitics_range

Specifies the range of parasitic resistance and capacitance values to model interconnect skew effects.

Syntax

status *set_parasitics_range*

```
[-resistance_factor {r_factor1 r_factor2}]
[-ground_capacitance_factor {c_factor1 c_facator2}]
[-coupling_capacitance_factor {cc_factor1 cc_factor2}]
[-capacitance_factor {cap_factor1 cap_factor2}]
[-layers  {layer_name1, layer_name2, …}]
[-correlation_factor corr_factor]
```

Data Types

```
r_factor1, r_factor2       float
c_factor1, c_factor2       float
cc_factor1, cc_factor2     float
```

```
layer_name1                 string
corr_factor          float
```

Arguments

`-resistance_factor {r_factor1, r_factor2}`

Two scaling factors to represent the min and max corners of parasitic resistance values. The factors are positive floating-point numbers. For example, {1.0 1.2} represents the resistance values can be increased from typical (1.0) to max corner (1.2). {0.8 1.2} represents the resistance values can be increased from min corner (0.8) to max corner (1.2).

`-ground_capacitance_factor {c_factor1, f_factor2}`

Two scaling factors for the min and max corner of net-to-ground capacitance values.

`-coupling_capacitance_factor {cc_factor1, cc_factor2}`

Two scaling factors for the min and max corner of cross-coupling capacitance values. If -ground_capacitance_factor is also set, coupling capacitances and ground capacitaces should use the same factors.

`-layers {layer_name1, layer_name2, …}`

Applies the parasitic scaling to the specified layers. Multiple input layers specified in the same set_parasitics_range command is designed for double patterning and they should be fully correlated.

`-correlation_factor corr_factor`

Set correlation coefficient for resistance and capacitance factors. Default is -1. If this option is applied, user must provide both resistance and capacitance factors.

Description

The *set_parasitics_range* command is perquisite for both interconnect skew bounding analysis and interconnect skew path analysis. This is a more accurate analysis to model the parasitic variations for min and max parasitic corners compared to scaled_parasitcs.

This command accepts three kinds of parasitic ranges:

o One-sided parasitic range: Parasitic range has 1.0 on one side and other positive values on the other side. This will model the parasitic effect between typical and min/max corners.

o Two-sided parasitic range: Parasitic range two different values that is greater than 1.0 on one side and lesser than 1.0 on the other side. This will model the parasitic effect between min and max corners.

o Single-valued parasitic range: Parasitic range has the same positive value that is not 1.0 on both side. In this setting, interconnect skew analysis will simply scale the parasitics.

To report or cancel scaling factors, use the *report_parasitics_range* or *reset_parasitics_range* command.

Examples

The following example model the interconnect skew impact from typical to CMAX corner with the resistance scaling factor 1.1, the ground capacitance and coupling capacitances scaling factor 0.92. This scaling applies to layer M1 and M1_mask1, M1_mask2.

```
pt_shell> set_parasitics_range -resistance_factor {1.0 1.1}

                  -capacitance_factor {1.0 0.92} -layers {M1 M1_mask1
 M1_mask2}
```

See Also

- read_parasitics
- reset_parasitics_range

---

set_path_margin

Specifies a margin to adjust required times for specified paths in the current design.

Syntax

status *set_path_margin*

```
[-rise]
[-fall]
[-setup]
[-hold]
[-reset_path]
[-increment]
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-through through_list
   | -rise_through rise_through_list
   | -fall_through fall_through_list]
[-sms_scenarios sms_scenarios_list]
[-domain_crossing domain_crossing_mode]
[-comment comment_string]
margin_value
```

Feedback

Data Types

comment_string	string
fall_from_list	list
fall_through_list	list
fall_to_list	list
from_list	list
margin_value	float
rise_from_list	list
rise_through_list	list
rise_to_list	list
sms_scenarios_list	collection
domain_crossing_mode	string
through_list	list
to_list	list

Arguments

`margin_value`

Specifies the value of the margin by which the required time is adjusted for the specified paths. A positive value results in a more restrictive or tighter check. A negative value results in a less restrictive or looser check. The value must be specified in the same units as those of the technology library used during optimization.

`-rise`

Applies the timing adjustment only to paths having a rising signal at the path endpoint.

`-fall`

Applies the timing adjustment only to paths having a falling signal at the path endpoint.

By default, paths having either a rising or falling at the endpoint are adjusted.

`-setup`

Applies the timing adjustment to only setup timing constraints for the specified paths.

`-hold`

Applies the timing adjustment to only hold timing constraints for the specified paths.

By default, both setup and hold constraints are adjusted.

`-reset_path`

Indicates that existing point-to-point exception information is to be removed from the specified paths. If used with the *-to* option only, all paths leading to

the specified endpoints are reset. If used with *-from* only, all paths leading from the specified startpoints are reset. If used with the *-from* and *-to* options, only the paths between those points are reset. Only information of the same rise or fall setup or hold type is reset. Using this option is equivalent to using the *reset_path* command with similar arguments before issuing the *set_path_margin* command.

`-increment`

When multiple set_path_margin commands are applied at different stages of the design without using the *-increment* option, only one path margin is applied to the path. With the*-increment* option it is possible to have multiple path margins applied to the path. There is a variable, *timing_enable_cumulative_incremental_path_margin*, to control the accumulation of all the incremental path margins. If there are multiple set_path_margin for the same path with the different precedence, the path margin with highest precedence gets applied. But, with this new *-increment* command option all the satisfied exceptions path margins are applied. If *set_path_margin* is set for all the nets in the path, this option helps to add all the path margins of the nets and applied to the same end point. The *report_timing -exceptions all* shows all the path_margins that were applied to the path.

`-from` *`from_list`*

Specifies a list of timing path startpoint objects. A valid timing startpoint is a clock, a primary input or inout port, a sequential cell, a clock pin of a sequential cell, a data pin of a level-sensitive latch, or a pin that has input delay specified. If a clock is specified, all registers and primary inputs related to that clock are used as path startpoints. If a cell is specified, one path startpoint on that cell is affected.

`-rise_from` *`rise_from_list`*

Same as the *-from* option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

`-fall_from` *`fall_from_list`*

Similar to the *-rise_from* option, except that the path must fall from the objects specified.

When the timing path has interior transparent latches, the *-from*, *-rise_from*, and *-fall_from* options refer to the clock, clock pin, or data pin of the last interior latch on the path, not to the initial startpoint of the timing path.

You can use only one of the *-from*, *-rise_from*, and *-fall_from* options.

`-to` *`to_list`*

Specifies a list of timing path endpoint objects. A valid timing endpoint is a clock, a primary output or inout port, a sequential cell, a data pin of a sequential cell, or a pin that has output delay specified. If a clock is specified, all registers and primary outputs related to that clock are used as path endpoints. If a cell is specified, one path endpoint on that cell is affected.

`-rise_to` *`rise_to_list`*

Same as the *-to* option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

`-fall_to` *`fall_to_list`*

Similar to the *-rise_to* option, but applies only to paths falling at the endpoint.

You can use only one of the *-to*, *-rise_to*, and *-fall_to* options.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, and nets through which the paths must pass for maximum delay definition. By default, a net is interpreted to imply its driver pins. In case the previous *through_list* includes the driver, the net is interpreted to imply its load pins. If you omit the *-through* option, all timing paths specified using the *-from* and *-to* options are affected.

`-rise_through` *`rise_through_list`*

Similar to the *-through* option, but applies only to paths with a rising transition at the specified objects.

`-fall_through` *`fall_through_list`*

Similar to the *-through* option, but applies only to paths with a falling transition at the specified objects.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this path margin is applied. When this option is not given the path margin applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

`-domain_crossing` *`domain_crossing_mode`*

Specifies one of the domain crosing modes for which this path margin is applied. The domain crossing modes are "only_crossing" and "exclude_crossing". If only_crossing option is used, then the path margin is applied only for cross domain paths. If exclude_crossing option is used, then the path margin is applied only for non-cross domain paths. When -domain_crossing option is not given, path margin applies for all the paths.

`-comment` *`comment_string`*

Associates a string description with the command for tracking purposes. This can be useful when writing out SDC to a tag, such as the methodology or tool that originally synthesized the command.

Description

This command adjusts the data required time for specified paths by a specified amount, *margin_value*. The required time for any startpoint in *from_list* to any endpoint in *to_list* is adjusted by *margin_value*. A positive margin value results in a more restrictive or tighter check, whereas a negative margin value results in a less restrictive or looser check. Based on the type of timing constraint, either setup or hold, the margin value is either subtracted from or added to the default data required time.

The *set_path_margin* command is a point-to-point timing exception command; that is, it adjusts the required time of the path and therefore the endpoint slack for the timing paths.

Other point-to-point exception commands include the *set_multicycle_path*, *set_min_delay*, *set_max_delay*, and *set_false_path* commands.

When a path has transparent latches in the interior of the path, *set_path_margin* is only applied when all the path specifiers are satisfied at or after the last interior latch on the path. The *-from* parameter refers to the clock, clock pin, or D pin of the last interior latch. The *-through* pins must be after the last interior latch.

If a path satisfies multiple timing exceptions, the following rules are applied to determine which exceptions take effect. Rules referring to *-from* apply equally to *-rise_from* and *-fall_from*, and similarly for the rise and fall options of *-through* and *-to*.

1. `set_path_margin -from` *`pin`* `-to` *`pin`*
2. `set_path_margin -from` *`pin`* `-to` *`clock`*
3. `set_path_margin -from` *`pin`*
4. `set_path_margin -from` *`clock`* `-to` *`pin`*
5. `set_path_margin -to` *`pin`*
6. `set_path_margin -from` *`clock`* `-to` *`clock`*
7. `set_path_margin -from` *`clock`*
8. `set_path_margin -to` *`clock`*

Note that if two or more exceptions have the same path options but differ only on the sense specified, then the most constraining value is used. An example is shown below:

1. `set_path_margin -from` *`pin`* `-to` *`pin`* `55`

```
2. set_path_margin -rise_from pin -to pin 60

3. set_path_margin -fall_from pin -to pin 50
```

In this example, all three exceptions will have the same precedence, however, the more constraining value will be used. Exception (1) -from is more constraining than exception (3) -fall_from path; and exception (2) -rise_from is more constraining than (1) -from path.

To report the path margin that applies to a given timing path, use the *report_timing -exceptions dominant* command.

Examples

The following example adjusts the required time to a port named OUT by 12 units. The adjustment results in a more restrictive, tighter timing constraint because the margin value is positive.

```
pt_shell> set_path_margin 12 -to {OUT}
```

The following example specifies that the required time of all paths from cell FF1 that pass through cell I1 and end at cell FF2 must be adjusted by 15.0 units.

```
pt_shell> set_path_margin 15 -from {FF1} -through {I1} -to {FF2}
```

The following example specifies that all paths to endpoints clocked by CLK must be adjusted by 10 units.

```
pt_shell> set_path_margin 10 -to [get_clocks CLK]
```

The following example specifies that required time of all the cross domain paths from cell FF1 and end at cell FF2 must be adjusted by 11.0 units.

```
pt_shell> set_path_margin 11 -from {FF1} -to {FF2} -domain_crossing
 only_crossing
```

The following example specifies that required time of all the paths from INFF2 and to OUTFF2 must be adjusted by 0.3. The incremental path margin 0.2 is added to the regular path margin 0.1.

```
pt_shell> set_path_margin 0.10 -from INFF2/CP -to OUTFF2/D
pt_shell> set_path_margin 0.20 -from INFF2/CP -to OUTFF2/D -increment
```

The following example specifies that the required time of all the paths from INFF2 and to OUTFF2 must be adjusted by 0.4. The incremental path margin 0.3 is added to the regular path margin 0.1. The last incremental path margin 0.3 overrides all the preceding ones.

```
pt_shell> set_path_margin 0.10 -from INFF2/CP -to OUTFF2/D
pt_shell> set_path_margin 0.20 -from INFF2/CP -to OUTFF2/D -increment
pt_shell> set_path_margin 0.30 -from INFF2/CP -to OUTFF2/D -increment
```

Feedback

The following example specifies that the required time of all the paths to the end point OUTFF2 must be adjusted by 0.6. If the -increment path margin values applied at through pins that satisfies the end point, it accumulates all those increment values and value of 0.6 is applied to the end point.

```
pt_shell> set_path_margin 0.10 -from INFF2/CP -to OUTFF2/D
pt_shell> set_path_margin 0.20 -through DPB1/YB -increment
pt_shell> set_path_margin 0.30 -through DPB2/YB -increment
```

See Also

- create_clock
- current_design
- group_path
- report_constraint
- report_path_group
- reset_design
- reset_path
- set_false_path
- set_input_delay
- set_min_delay
- set_max_delay
- set_multicycle_path
- set_output_delay
- timing_enable_cumulative_incremental_path_margin

---

set_pin_abstraction

Specifies the pins to include in a block abstraction.

Syntax

```
status set_pin_abstraction
  [-keep]
  [-clear]
  [-list]
  [pin_list]
```

Feedback

Data Types

pin_list list

Arguments

-keep

Includes the specified pins in the block abstraction generated at HyperScale block-level analysis and visible at top-level analysis.

-clear

Clears all pins with user-specified abstraction settings.

-list

Reports pins with user-specified abstraction control.

pin_list

Specifies a list of pins. Either a collection of pins or a list of patterns that match pins in the current design.

Description

This command customizes the HyperScale block abstraction behavior for specific pins.

This command is only effective in block-level HyperScale runs. The pins specified are considered by subsequent *update_timing*.

By default, HyperScale block-level analysis makes decisions based on the block interface topology and timing characteristics to retain interface portion of the circuit and make them visible for top-level analysis to achieve the best runtime and memory.

This command allows you to retain and keep specific pins from HyperScale block abstraction, such that these pins become visible at top-level analysis.

For each pin specified in the list, PrimeTime keeps the exact pin and the net connected with it, plus all the leaf pins on the same net to make the net complete. The actual objects retained is strictly based on local physical wire connectivity. There are no forward or backward path or logic tracing from these pins to span the visibility to additional objects.

Examples

This example keeps all the pins in the fanout of myReg in the block abstraction:

```
pt_shell> set_pin_abstraction -keep [all_fanout -from myReg/Q]
```

Feedback

See Also

- collections
- set_port_abstraction
- update_timing
- hier_enable_analysis

---

set_placement_spacing_label

Sets an inter-cell spacing constraint label on one or more reference cells in preparation for defining minimum spacing rules between library cells.

Syntax

status *set_placement_spacing_label*

```
-name label_name
-side (left|right|both)
-lib_cells fIlist_of_lib_cells
[-row cell_row]
```

Data Types

```
label_name          string
list_of_lib_cells   list
cell_row            integer
```

Arguments

`-name label_name`

Inter-cell constraint label to assign to one or more reference cells.

`-side (left|right|both)`

Specifies that the given list of reference cells have the given label assigned to the left side, the right side, or both sides of the cell, when viewed in the "north" orientation.

`-lib_cells list_of_lib_cells`

Specifies a list of reference cells that have the given label assigned to them on one or both sides. The argument must be a list of library cell names.

`-row cell_row`

Specifies a row (starting at 1) upon which the given reference cells are assigned the given label. By default, the label is assigned to all rows of a multi-height cell.

Feedback

Description

This command assigns a label to a list of reference (library) cells to set inter-cell spacing constraints to be enforced by the legalization tool. A label is a string that you assign to one or both sides of a standard cell.

This command works together with the *set_placement_spacing_rule* command, which defines the illegal range of spacing between two adjacent labels. All spacing rules specified by *set_placement_spacing_rule* command must be satisfied to produce a legal placement.

With inter-cell spacing rules specified, standard cells can be placed next to each other only when they do not violate the spacing rules.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of standard cells. The labels and spacing rules are stored in the library and are applied to all designs using the library.

Examples

The following commands assign the label "X" to reference cells AND* on the left side of the cell, and to reference cells NAND* on the right side of the cell (for cells in the "North" orientation).

prompt> set_placement_spacing_label -name X -side left -lib_cells AND* prompt> set_placement_spacing_label -name X -side right -lib_cells NAND*

See Also

- set_placement_spacing_rule

---

set_placement_spacing_rule

Sets an intercell spacing constraint between reference cells that have been assigned labels with the set_placement_spacing_label command.

Syntax

status *set_placement_spacing_rule*

```
-labels list_of_label_names
min_max_spec
```

Data Types

```
list_of_label_names          list of two strings
min_max_spec                 list of two integers
```

Feedback

Arguments

`-labels list_of_label_names`

Specifies a list of exactly two intercell constraint labels previously defined by the set_placement_spacing_label command.

min_max_spec

Specifies the range of illegal x-spacing values in units of sites. *min_max_spec* follows the format *{minx maxx}* where minx <= maxx. The spacing between the labeled edges cannot be in the range from minx to maxx placement sites.

Description

This command assigns intercell spacing constraints between reference cells that have been assigned labels with the set_placement_spacing_label command. Cells with the given labels are prohibited from being placed adjacent to one another with a spacing between minx and maxx. The values are specified in units of placement sites.

Given two adjacent standard cells with an empty space in between, each label from the right side of the left cell is checked against each label from the left side of the right cell to see if any spacing rules are violated. Such pairwise checking of labels is applied to all adjacent cells to ensure placement legality.

A label typically represents some layout features inside a standard cell. A label is associated with the left or right boundary of a standard cell in the "North" orientation. When a cell is flipped, its associated labels are also flipped.

With intercell spacing rules specified, standard cells can be placed next to each other only when they do not violate the spacing rules.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of standard cells. The labels and spacing rules are stored in the library and are applied to all designs using the library.

Examples

In the following example, cells with labels "X" and "Y" are prohibited to abut (a spacing of 0 is prohibited).

```
prompt> set_placement_spacing_rule -labels {X Y} {0 0}
```

In the following example, cells with labels "X" are prohibited to be spaced within range of 3 to 5 placement sites.

```
prompt> set_placement_spacing_rule -labels {X X} {3 5}
```

See Also

- set_placement_spacing_label

Feedback

set_port_abstraction

Specifies the port fanin and fanout abstraction.

Syntax

```
status set_port_abstraction
  [-ignore]
  [-keep]
  [-clear]
  [-list]
  [port_list]
```

Data Types

port_list list

Arguments

`-ignore`

Indicates that the specified ports in the *port_list* whose fanout and/or fanin logic are pruned from the block abstraction generated at HyperScale block level analysis and not visible at top level analysis.

`-keep`

Indicates that the specified ports in the *port_list* whose fanout and/or fanin logic are included in the block abstraction generated at HyperScale block level analysis and visible at top level analysis.

`-clear`

Clears all the ports with user-specified abstraction settings. It clears both *-keep* and *-ignore* ports.

`-list`

Reports the ports with user-specified abstraction control.

port_list

Specifies a list of ports. Either a collection of ports or a list of patterns that match ports in the current design.

Description

This command specifies the HyperScale block abstraction behavior for specific ports and their fanout or fanin logic.

This command is only effective in block level HyperScale runs. The specified ports are considered by subsequent *update_timing* commands.

Feedback

By default, HyperScale block level analysis makes decisions of whether and how much a given port's fanin/fanout logic is retained based on the block interface topology and timing characteristics. If the port's fanin or fanout logic is retained, that interface portion of the circuit becomes completely visible for top level analysis, otherwise, only reduced visibility is expected for these ports at top level. These internal decisions are made to achieve the best runtime and memory.

This command allows you to customize HyperScale block abstraction behavior. Either force retain (with -keep) or remove (with -ignore) specific ports and their fanin/fanout logic in the HyperScale block abstraction, such that these ports become visible (with -keep) or reduced (with -ignore) at top level analysis.

For each port specified to keep, PrimeTime traces the fanin and fanout logic of the port and keep all the pins, cells and nets up to the first register (if input port) or from the last register (if output port) connected.

Examples

This example specifies that the fanout logic for ports SCAN* to be retained:

```
pt_shell> set_port_abstraction -keep [get_ports SCAN*]
```

This example specifies that the fanout logic for ports RST* be reduced or pruned:

```
pt_shell> set_port_abstraction -ignore [get_ports RST*]
```

This example removes all the user ignored or kept ports for custom abstraction:

```
pt_shell> set_port_abstraction -clear
```

See Also

- collections
- update_timing
- hier_enable_analysis

---

set_port_attributes

Sets the specified attributes and their values on port objects.

Syntax

string *set_port_attributes*

```
[-ports port_list]
[-elements element_list]
[-applies_to inputs | outputs | both]
[-driver_supply supply_set_ref]
```

```
[-receiver_supply receiver_supply_set_name]
[-repeater_supply repeater_supply_set_name]
[-attribute name_value_pair]
[-is_analog]
[-literal_supply]
[-model model_name]
```

Data Types

port_list	list
element_list	list
supply_set_ref	string
receiver_supply_set_name	string
repeater_supply_set_name	string
name_value_pair	string
is_analog	boolean
literal_supply	string
model_name	string

Arguments

`-ports` *port_list*

Specifies the collection of ports on which the attributes must be set. Wildcards are accepted in port names.

`-elements` *element_list*

Specifies a set of ports on the interface to the specified elements, excluding any supply ports. A design element reference '.' is accepted to designate the current scope.

`-applies_to` *inputs* | *outputs* | *both*

Specifies whether the given *set_port_attributes* command applies to input ports, output ports, or both for the specified elements. This option must be used in conjunction with the *-elements* option. The default is *both*.

`-attribute` *name_value_pair*

Specifies the name and value of the attribute to be set on the ports specified by the *-ports* option.

This option can be repeated multiple times to specify multiple attributes for the same set of ports in a single command.

`-driver_supply` *supply_set_ref*

Specifies the supply of the logic driving the port.

`-receiver_supply` *receiver_supply_set_name*

Specifies the supply of the logic reading the port.

`-repeater_supply` *repeater_supply_set_name*

Specifies the supply set used by a repeater driving the port.

`-is_analog`

Applies the *is_analog* attribute to mark the port as an analog port.

`-literal_supply`

Specifies the supply set used to implement a literal constant value associated with an input port instance. This option is read and ignore in PrimeTime.

`-model`

Specifies a module or library cell on whose ports the attribute is to be applied.

Description

This command sets the attributes and their value on a list of ports specified by the *-ports* option or on a list of instances specified by the *-elements* option.

Attributes can be set multiple times on the same ports. The last-applied value is used.

PrimeTime suite tools only support limited number of attributes. Unsupported attributes are ignored and a warning message is issued.

The *-receiver_supply* and *-driver_supply* port attributes can only be specified on the top-level ports and design. A warning is issued if non top-level design element is specified for the *-elements* option for the *-driver_supply* or *-receiver_supply* option. Specifications on invalid design elements are ignored.

The *-model* option can be used with *-is_analog* attribute.The *-is_analog* option is allowed to be used with or without *-model*.

The tool interprets a port attribute specified on a design element as less specific than one specified on a particular port of the design element.

Examples

The following example sets SS1 as the *driver_supply* on top-level input port IN1 and sets SS2 as the *receiver_supply* on top-level output port OUT1.

```
prompt> set_port_attributes -ports {IN1} \\
                -driver_supply SS1
prompt> set_port_attributes -ports {OUT1} \\
                -receiver_supply SS2
```

The following example sets SS1 as the *driver_supply* on top-level input ports and sets SS2 as the receiver_supply *on top-level output ports using -elements {.} -applies_to. '.' refers to the current scope, which is set to the top-level design in this case.*

Feedback

```
prompt> set_port_attributes -elements {.} \\
              -applies_to inputs \\
       -driver_supply SS1
prompt> set_port_attributes -elements {.} \\
              -applies_to outputs \\
              -receiver_supply SS2
```

The following example sets the *snsp_derived* attribute to *true* on ports power_port and ground_port in block Mult.

```
prompt> set_port_attributes -ports {Mult/power_port Mult/ground_port}
 -attribute iso_sink SS1
1
```

See Also

- create_supply_set

---

set_port_fanout_number

Sets the number of external fanout points on ports.

Syntax

string *set_port_fanout_number*

```
[-min]
[-max]
fanout_number
port_list
```

Data Types

fanout_number	int
port_list	list

Arguments

`-min`

Specifies the value for minimum condition.

`-max`

Specifies the value for maximum condition.

fanout_number

Specifies the number of external fanout points (Range: 0 to 100000).

port_list

Specifies a list of ports. Each element in the list is either a collection of ports or a pattern that matches ports on the current design.

Description

Sets the number of external fanout pins for ports in the current design. The number of pins is used (along with wire load models) to calculate capacitance and resistance of nets.

To remove port fanout number values, use the *remove_port_fanout_number* command. To show port fanout number information, use the *report_port -wire_load* command.

Examples

This example sets the external wire load model for ports OUT1* to 70x70 and specifies an external fanout number of 2.

```
pt_shell> set_wire_load_model 70x70 [get_ports OUT1*]
pt_shell> set_port_fanout_number 2 [get_ports OUT1*]
```

See Also

- collections
- remove_port_fanout_number
- report_port
- set_wire_load_model

---

set_postsynth_options

Specify RTL Architect commands to be executed after compilation is done.

Syntax

int *set_presynth_options*

```
[-file file_name]
[-cmds cmd_list]
```

Data Types

```
file_name       string
cmd_list        list
```

Arguments

-file *file_name*

Specifies tcl file which contains tcl commands to be executed after compile.

Feedback

`-cmds` *`cmd_list`*

if number of tcl commands is small, specify commands directly instead of creating separate file.

Description

The *set_postsynth_options* command pass RTL Architect tcl commands to be executed after compile. Use this command for reporting purpose.

Examples

The following example shows how to pass arbitrary command

```
pwr_shell> set_postsynth_options -cmds {
              puts "*** after compile ***"
              report_qor
           }
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_phy_tech_file
- set_synth_output_dir
- set_rtl_architect_shell
- set_presynth_options
- set_synth_options

---

set_power_analysis_options

Sets the options for power analysis.

Syntax

integer *set_power_analysis_options*

```
[-static_leakage_only]
[-through_mode]
[-waveform_interval sampling_interval]
[-cycle_accurate_cycle_count cycles]
[-cycle_accurate_clock clock]
[-waveform_format fsdb | out | none]
[-waveform_output file_prefix]
```

```
[-multi_rail_waveform supply_net_list]
[-include top | all_without_leaf | all_with_leaf]
[-include_groups group_list]
[-cells cell_list]
[-cycle_power_hierarchy cell_list]
[-sdpd_tracking enabled | disabled]
[-sdpd_tracking_cells cell_list]
[-separate_power_waveform leakage | glitch | all]
[-conflict_activity_cells cell_list]
[-exclude_arc_scaling_libcells libcell_list]
[-npeak npeak_size]
[-npeak_out file_prefix]
[-peak_power_instances]
[-propagate_through_feedbackloop]
```

Data Types

```
quantile              float
sampling_interval     float
cycles                integer
clock                 string
file_prefix           string
supply_net_list       list
group_list            list
cell_list             list
libcell_list          list
npeak_size                integer
```

Arguments

`-static_leakage_only`

Specifies that averaged power analysis will only calculate static leakage power and skip dynamic power calculation. This option applies only to averaged power analysis.

`-through_mode`

Enables through_mode propagation of clock activity throughout the clock network. When not specified, regular probabilistic propagation takes place in the clock networks as well. This option applies only to averaged power analysis.

`-waveform_interval`

Specifies the sampling interval that is used for power waveform. The default value is the timescale from the VCD file. This option applies only to time_based power analysis. The option will be ignored if VCD comes from RTL or zero-delay simulation.

`-cycle_accurate_cycle_count`

Specifies the number of clock cycles over which the power consumed by the design is considered constant when VCD is RTL or zero-delay. The default is 1.

This option applies only to time_based power analysis, specifically for RTL or zero-delay VCD or its equivalents (such as, VPD, etc.).

`-cycle_accurate_clock`

Specifies the reference clock for time_based power analysis, when VCD is RTL or zero-delay. The power consumed by the design is considered constant over one or more cycles of the specified clock. The default is the fastest clock in the design. This option applies only to time_based power analysis, specifically for RTL or zero-delay VCD or its equivalents (such as, VPD etc.).

`-waveform_format`

Specifies the format of the file in which the waveform data is to be written. The value for this option can be: *fsdb*, *out* or *none*. The default is *fsdb*. If the format is set to none, waveform data is not to be dumped, but peak power is still calculated. This option applies only to time_based power analysis.

`-waveform_output`

Specifies the prefix of the file in which the waveform data is to be written. The default is *primetime_px*. This option applies only to time_based power analysis.

`-multi_rail_waveform`

Specifies the list of supply nets for which the waveform data is to be written. Separate waveforms will be generated for each of these rails. You need to enable multirail analysis to use this feature. This option applies only to time_based power analysis. By default, multirail wave generation is disabled.

`-include`

Specifies the objects in the design hierarchy to be monitored for waveform generation. The tool will only be able to show the power waveforms limited by this option. There are three allowed values for the option. If the value is *top*, only the top-level design is monitored for power waveform generation. If the value is *all_without_leaf*, all design hierarchies except leaf cells are monitored. If the value is *all_with_leaf*, all design hierarchies including leaf cells are monitored. The default value is *all_without_leaf*. This option applies only to time_based power analysis.

`-include_groups`

Specifies the power groups to be monitored for power waveform generation. When specified, the tool shows the power waveforms only for the specified power groups. By default, no power group is monitored. This option applies only to time_based power analysis.

`-cells`

Specifies the cell names or collections of cells for which power needs to be calculated. By default, PrimePower calculates power for the whole design.

`-propagate_through_feedbackloops`

Enables activity propagation outside of power analysis cells through feedback loops.This option can only be used with -cells option.

`-cycle_power_hierarchy`

Specifies the names or collections of hierarchical cells for which cycle based power reports will be created from Cycle Power. By default, Cycle Power only outputs cycle based power reports for the whole design.

`-sdpd_tracking`

Enables or disables SDPD activity tracking for cells specified in the -sdpd_tracking_cells list. The default is *disabled*.

`-sdpd_tracking_cells`

Specifies the list of cells for which SDPD activity is to be tracked. The default list contains all black box and memory cells.

`-separate_power_waveform`

Enables generation of separate leakage, glitch and all power waveforms. This option can be used with -multi_rail_waveform option. In this case, separate waveforms will be generated for each supply net from the given list of supply nets. This option applies only to time_based mode.

`-conflict_activity_cells`

Specifies the list of cells for which activity in the VCD/FSDB file has conflicts with the activity of the parent cells. A conflict activity for a given net is defined as activity with different transitions in the same time stamp. A net can have several net segments which are across hierarchical levels. For these nets, the user can generate separate VCD/FSDB files which can be later merged together into a single FSDB file. The activities for different net segments of such a net can be different across hierarchical levels. This option helps to define the list of conflict cells with such conflicting activity and tries to resolve them.

In order to enable this feature, set the *power_enable_merged_fsdb* variable to true. This feature works in both averaged and time-based analysis modes. For time-based mode, this feature only works with gate-level VCD/FSDB files which do not need any activity propagation.

`-exclude_arc_scaling_libcells`

Specifies the list of library cells for which internal power arc would not be scaled.

`-npeak`

Reports N worst power values of the design in time_based mode. The allowed values are from 2 to 100. When specified, the tool generates a separate report

which contains the N worst power values of the design and their corresponding time.

`-npeak_out`

Specifies the prefix of the npeak report to be generated. If this option is not specified, by default the report file is named *primepower_npeak.out*.

`-peak_power_instances`

Reports instances which contribut to N worst power values of the design waveform in time_based mode. This option works only with the -npeak option; the allowed values are from 2 to 10. When used together with the -npeak option, the tool generates a separate report which contains the N worst power values of the design, corresponding time and design instances contributing to N worst power values. Use the -npeak_out option to specify the name of the generated instance report. If the -npeak_out option is not specified, by default the report file is named *primepower_npeak_instances.out*.

Description

There are several power analysis modes in PrimePower, such as *averaged, time_based, or leakage_variation. The mode is specified by the power_analysis_mode variable before the key power analysis commands. The working command for all power analysis mode is update_power. The update_power command itself does not take any option. The set_power_analysis_options command is used between the power_analysis_mode variable and the update_power command to specify various options for different power modes in PrimePower. The options from this command will control how power analysis of the specified mode behaves.*

Each option may be used for one or more modes of power analysis. If an option is not supposed to be used in a power mode, the command stops and issues an error message. The *set_power_analysis_options* command without any option resets all the power analysis options to the default values. Rerunning the *set_power_analysis_options* command overwrites the previous configuration and resets the rest of power analysis options to the default values. The -cycle_accurate_clock and -cycle_accurate_cycle_count options are exclusive with the -waveform_interval option in this command.

The following table lists the options supported in each of the power modes.

```
------------------------------+----------+------------+-----------------
-------+-------------------+
 |                              |          |            |       time_based
       |                     |
 |                              | averaged | time_based | (RTL or zero
 delay vcd) | leakage_variation |
 ------------------------------+----------+------------+-----------------
--------+-------------------+
 | -static_leakage_only         |   Yes    |     No     |            No
       |          No         |
```

Feedback

-through_mode	Yes	No	No
	No		
-variation_quantile	No	No	No
	Yes		
-waveform_interval	No	Yes	No
	No		
-cycle_accurate_cycle_count	No	No	Yes
	No		
-cycle_accurate_clock	No	No	Yes
	No		
-waveform_format	No	Yes	Yes
	No		
-waveform_output	No	Yes	Yes
	No		
-multi_rail_waveform	No	Yes	Yes
	No		
-include	No	Yes	Yes
	No		
-include_groups	No	Yes	Yes
	No		
-cycel_power_hierarchy	No	No	Yes
	No		
-cells	Yes	Yes	Yes
	Yes		
-sdpd_tracking	Yes	Yes	Yes
	No		

```
 ------------------------------+----------+------------+-----------------
--------+--------------------+
 | -sdpd_tracking_cells        |   Yes    |    Yes     |           Yes
        |          No        |
 ------------------------------+----------+------------+-----------------
--------+--------------------+
 | -conflict_activity_cells    |   Yes    |    Yes     |            No
        |          No        |
 ------------------------------+----------+------------+-----------------
--------+--------------------+
 | -npeak                      |    No    |    Yes     |           Yes
        |          No        |
 ------------------------------+----------+------------+-----------------
--------+--------------------+
 | -npeak_out                  |    No    |    Yes     |           Yes
        |          No        |
 ------------------------------+----------+------------+-----------------
--------+--------------------+
```

When SDPD tracking is enabled, SDPD (state-dependent and path-dependent) switching activity for the sepcified cells will be tracked during power analysis. So later when the *write_saif* command is performed, the resulting SAIF file has the SDPD switching activity data.

Examples

The following example runs time_based power analysis. The time interval is 10 ns. The waveform output file is named my_design.fsdb.

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd -strip_path this_strip_path my_design.vcd
pt_shell> set_power_analysis_options -waveform_interval 10
 -waveform_output my_design
pt_shell> update_power
```

The following example runs averaged power analysis. Only leakage power is calculated for fast turn-around time.

```
pt_shell> set power_analysis_mode averaged
pt_shell> set_power_analysis_options -static_leakage_only
pt_shell> update_power
```

The following example runs averaged power analysis. This will enable the pass-through propagation of toggle rates in the clock network.

```
pt_shell> set power_analysis_mode averaged
pt_shell> set_power_analysis_options -through_mode
pt_shell> update_power
```

The following example runs time_based power analysis for an RTL-based design. The reference clock is *clk*.

```
pt_shell> set power_analysis_mode time_based
pt_shell> source name_mapping_info.tcl
pt_shell> read_vcd -rtl -strip_path this_strip_path my_rtl_design.vcd
pt_shell> set_power_analysis_options -cycle_accurate_clock clk
pt_shell> update_power
```

The following example will cause PrimePower to do leakage power variation analysis. The variation quantile is *0.9*.

```
pt_shell> set power_analysis_mode leakage_variation
pt_shell> set_power_analysis_options -variation_quantile quantile
pt_shell> update_power
```

The following example resolves conflict activity on a list of cells. This option needs to be set before reading the merged VCD or FSDB file. This feature works in average and time_based power modes (with a gate-level VCD or FSDB file).

```
pt_shell> set power_enable_merged_fsdb true
pt_shell> set_power_analysis_options -conflict_activity_cells { u0 }
pt_shell> read_vcd -strip_path this_strip_path my_design.vcd
pt_shell> update_power
pt_shell> report_power -cell -leaf
```

See Also

- power_analysis_mode
- report_power_analysis_options
- update_power
- report_power

---

set_power_base_clock

Specify power_base_clock for a given set of objects.

Syntax

integer *set_power_base_clock*

```
[-clock_name name_string]
[object_list]
```

Data Types

name_string	string
object_list	list

Feedback

Arguments

`-clock_name` *name_string*

Specify the clock name.

object_list

Specify a list of objects in the design. Only nets, pins and ports are supported.

Description

For a given net/pin activities are applied by specifying toggle rates relative to it's related clock (power_base_clock) . Currently for every pin, power_base_clock is identified by PrimePower based on design constraints and timing paths through the pin. This command allows PrimePower to use the specified clock as the power_base_clock of a given pin/net instead of the implied one.

This needs to be specified before any set_switching_activity command and prior to 'update_power' . The actual association is done during 'set_switching_activity / 'update_power'.

See Also

- power_enable_clock_scaling

---

set_power_budget

Sets target power budget on hierarchical instances in the design.

Syntax

```
int set_power_budget
budget_value
[-cell cell_list]
[-rails rail_list]
[-propagate]
[-exclude exclude_list]
[-exclude_group exclude_group_list]
[-exclude_rail exclude_rail_list]
[-cell_group cell_group_list]
[-power_type power_type_list]
```

Data Types

```
cell_list           list
rail_list           list
exclude_list        list
exclude_group_list  list
cell_group_list     list
exclude_rail_list   list
```

```
power_type_list     list
budget_value        float
```

Arguments

budget_value

Specifies the power budget that is applied on the specified hierarchical instances.

-cell *cell_list*

Specifies a list of hierarchical instances on which the specified power budget is applied.

-rails *rail_list*

Specifies a list of supply nets for applying supply-net-specific power budget on the specified hierarchical instances.

-propagate

Specifies derived scale factor upon setting the power budget on a supply net, to propagate to all the child supply net segment of this supply net.

-exclude *exclude_list*

Specifies a list of cells to exclude from power budget getting applied on them.

-exclude_group *exclude_group_list*

Specifies a list of power groups to exclude from power budget getting applied on them.

-exclude_rail *exclude_rail_list*

Specifies a list of supply_nets to exclude from power budget getting applied on them.

-cell_group *cell_group_list*

Specifies a list of power groups under hierarchies specified by -cell option on which power budget is applied. Must be used with -cell option.

-power_type *power_type_list*

Specifies a list of power types on which power budget is applied. Valid values are internal, switching and leakage.

Description

The command sets power budget on a list of hierarchical instances in the design. If the list is not specified, power budget is applied on the design. The specified power budget and actual computed power for the hierarchical instance would be used to calculate a scaling factor to scale power for all the instance in the hierarchical block, so that the

scaled power matches the specified power budget for the instance. With power budget set on the design, the *report_power* command would report the scaled power for the design.

Total power budget applied on the hierarchical instance must not be smaller than the calculated actual total leakage power for the hierarchical instance.

Supply-net-specific power budget can also be applied on the hierarchical blocks in which case power associated with the supply net would only be scaled.

Budget can be applied across multiple hierarchy levels; precedence increases with moving down the hierarchy levels in the design. Setting power budget on a hierarchical instance overrides the previously set budget, if any.

Use the *reset_power_budget* command to clear the applied budgets on the design.

Examples

The following example sets a power budget of value 1.135e-04 on the hierarchical instance four_mini_inst/two_mini_1/a.

```
pt_shell> set_power_budget 1.135e-04 -cell four_mini_inst/two_mini_1/a
```

The following example sets a power budget of value 9.354e-05 on the rail four_mini_inst/VDD_MINI_1 for the hierarchical instance four_mini_inst/two_mini_1/a.

```
pt_shell> set_power_budget 9.354e-05 -cell four_mini_inst/two_mini_1/a
 -rails four_mini_inst/VDD_MINI_1
```

See Also

- report_power
- get_attribute
- get_power_per_pgpin
- report_power_budget
- reset_power_budget

---

set_power_check_attributes

Allows user to specify whether attribute pin_propagation_check, pin_qm_propagation_check, internal_power_check, leakage_power_check will be computed or not. User must set variable power_enable_check_attributes to true for this command to work.

Feedback

Syntax

```
string set_power_check_attributes
[-pin_propagation_check]
[-pin_qm_propagation_check]
[-internal_power_check]
[-leakage_power_check]
[-all]
```

Arguments

`-pin_propagation_check`

This option must be provided for attribute pin_propagation_check to get computed.

`-pin_qm_propagation_check`

This option must be provided for attribute pin_qm_propagation_check to get computed.

`-internal_power_check`

This option must be provided for attribute internal_power_check to get computed.

`-leakage_power_check`

This option must be provided for attribute leakage_power_check to get computed.

`-all`

This option must be provided for attribute pin_propagation_check , pin_qm_propagation_check , internal_power_check and leakage_power_check to get computed.

Description

This command enables user to select which attribute to be computed from pin_propagation_check, pin_qm_propagation_check , internal_power_check , leakage_power_check or all attributes.

Examples

The following *set_power_check_attributes command only enable pin_propagation_check.*

```
pt_shell> set_power_check_attributes -pin_propagation_check
```

The following *set_power_check_attributes command enable pin_propagation_check and pin_qm_propagation_check.*

```
pt_shell> set_power_check_attributes -pin_propagation_check
 -pin_qm_propagation_check
```

The following *set_power_check_attributes command enable pin_propagation_check , pin_qm_propagation_check, internal_power_check and leakage_power_check.*

```
pt_shell> set_power_check_attributes -all
```

See Also

- report_power_check_attributes

---

set_power_clock_scaling

Specify clock clock scaling for power analysis.

Syntax

integer *set_power_clock_scaling*

```
[-period period_value]
[-ratio ratio_value]
[clock_objects]
```

Data Types

period_value	float
ratio_value	float
clock_objects	list

Arguments

`-period period_value`

Specify the period of clock which is used in simulation.

`-ratio ratio_value`

Specify the clock ratio (period_used_in_VCD_or_SAIF / period_used_in_SDC).

clock_objects

Specify a list of clocks in the design.

Description

Clock frequency used in value change dump (VCD) and Switching Activity Interchange Format (SAIF) files, from logic simulation, for functional verification can differ from the clock frequency used in Synopsys Design Constraints (SDC) file for timing and power analysis. This command tells PrimePower what frequency is used in logic simulation so that PrimePower can scale averaged power numbers properly.

To enable this feature, set the *power_enable_clock_scaling* variable to *true*.

The clock_objects option in this command is a list of clocks in the design that have the specified period or ratio values used in simulation for VCD or SAIF generation. Here the ratio is defined as period-in-simulation / period-in-analysis. The command can be used multiple times, once for each clock. If the same clock is used more than once, a warning message is issued and the clock specified last takes effect. The *-period* and *-ratio* options cannot be used simultaneously. If no clock object is specified, only the ratio option is allowed and the ratio value is applied to all the nets that do not have an associated simulation clock.

The tool only scales the switching activities from the VCD file or the SAIF file and activities that are propagated or implied. The actual scaling for switching activity and averaged power analysis is performed by the tool during the execution of the update_power command. You can also check the power_base_clock attribute on the nets to identify the clock that controls the activity scaling of the net. The command returns 1 on success and 0 on failure.

To perform repeated scaling, you need to re-specify the scaling factor. The scaling factor is applied w.r.t the original toggle rate . The tool keeps track of the factor by which the toggle rates are already scaled. For example, if a net has an original toggle rate 1.0 and the *-ratio* option of the *set_power_clock_scaling* command is set to 2, the 1st scaling results in the net's toggle rate to 2 (the original toggle rate 1.0 is replaced). To further scale the toggle rates by twice , you need to specify a scaling factor of 4. To come back to the original toggle rates, you can use *set_power_clock_scaling 1* .

The clock scaling is done only for averaged power. This feature cannot be applied to a time-based peak power analysis.

Examples

A sample script is listed below. In this case, duration logic simulation the SAIF file mac.saif is generated with clock period 24, so you use set_power_clock_scaling period 24 [get_clock clk], and you specify SDC clock in mac.sdc (it could be period 12, for example). Report_power will report the power consumption when the design is operated at SDC clock period 12, rather than logic simulation clock period 24.

```
set power_enable_analysis true
set power_analysis_mode averaged
set search_path         "../src/hdl/gate ../src/lib/snps . "
set link_library        " * core_typ.db"

read_verilog            mac.vg
current_design          mac
link
read_sdc ../src/hdl/gate/mac.sdc
read_parasitics ../src/annotate/mac.spef.gz
read_saif "../sim/mac.saif" -strip_path "tb/macinst"

set_power_clock_scaling -period 24 [get_clock clk]
set power_enable_clock_scaling true
```

```
update_power
report_power

quit
```

See Also

- power_enable_clock_scaling

---

set_power_delay_shifted_event_analysis_options

Sets the options for delay shifted event analysis.

Syntax

integer *set_power_delay_shifted_event_analysis_options*

```
[-effort_level effort_string]
[-write_activity_file output_file_name]
[-include_external_delay ]
[-glitch_pulse_r pulse_ratio]
[-zero_delay_power ]
[-zero_delay_report_type report_type_string]
[-rtl]
[-nets report_type_string]
[-exclude_nets report_type_string]
[-max_fanout report_type_int]
[-max_slew  pulse_ratio]
[-disable_shifting_through_power_switch ]
```

Data Types

```
effort_string          string
output_file_name       string
pulse_ratio            float
report_type_string     string
report_type_int        int
```

Arguments

`-effort_level`

Specifies the effort that is used in shifting the events using the delays obtained from timer (PrimeTime). Valid values are *low* and *high*. The default is *high*.

`-exclude_nets`

Specifies the nets that is excluded from delay shifting. Wire delay is taken to be zero for delay shifting purpose. For the driver and load cell of the net ,input and output arc is capped at max cell delay.

`-max_fanout`

Specifies max_fanout limit for net to be considered as high fanout net for delay shifting purpose.

`-max_slew`

Specifies max_slew limit for net to be considered as high fanout net for delay shifting purpose.

`-write_activity_file`

Generates a Gate-level delay annotated activity file (in the FSDB format) during power analysis.

`-include_external_delay`

Includes external delays for primary input ports in event shifting. By default, external delays are not used in event shifting.

`-rtl`

When option '-rtl' is specified then activity files contains rtl objects only. This option is applicable only when option 'write_activity_file' is provided by user.

`-nets`

When option '-nets' is specified then activity files contains nets objects provided by user. This option is applicable only when option 'write_activity_file' is provided by user.

`-glitch_pulse_r`

Filters out glitch pulses on a cell whose width is less than the specified percentage of the cell delay. By default, this value is 100, meaning all glitch pulses whose width is greater than the cell delay is propagated.

`-zero_delay_power`

Runs zero-delay power analysis simultaneously in the background, along with "delay-shifted event analysis". At the end of the analysis, a power report, named report_power_zd.txt, is generated for zero-delay analysis.

Redirection of log from update_power command is not supported with option *-zero_delay_power*.

`-zero_delay_report_type`

Specifies the power report type for background zero-delay power analysis. Three file types are supported: summary, cell_power, and net_power. The default is summary.

`-disable_shifting_through_power_switch`

Disables delay shifting through power switch cells.

Feedback

Description

This command is used when the *power_enable_delay_shifted_event_analysis* variable is set to true. The command is used between *power_enable_delay_shifted_event_analysis* and *update_power*. The options from this command control how delay aware peak power analysis uses RTL or zero-delay VCD/FSDB.

The delay as obtained from timer is added to each event time in the zero-delay input FSDB/VCD. Starting from each register output, the tools adds the delays and accumulated along the combinational path and continues till another register is found in the path where the delay is reset. Latches and clock gating cells (ICGs) are also included in this path between two registers.

For "low" effort level, the tool uses the worst path delay through a cell output to shift the event of the given cell output. For "high" effort level, the tool chooses the path that is sensitized by the input FSDB/VCD at each timestamp and uses the delay along this path to shift the events. This option has a high runtime overhead but is expected to give better accuracy.

When the -include_external_delay option is specified, the tool uses the delay specified by *set_input_delay* to shift events for primary input ports.

Examples

The following example runs peak power analysis with delay shifted events.

```
pwr_shell> set power_analysis_mode time_based
pwr_shell> set power_enable_delay_shifted_event_analysis true
pwr_shell> read_vcd -rtl -strip_path this_strip_path my_design.vcd
pwr_shell> update_power
```

The following example runs peak power analysis with delay shifted events and generates a delay-annotated gate-level activity file named shifted_activity.fsdb.

```
pwr_shell> set power_analysis_mode time_based
pwr_shell> set power_enable_delay_shifted_event_analysis true
pwr_shell> read_vcd -rtl -strip_path this_strip_path my_design.vcd
pwr_shell> set_power_delay_shifted_event_analysis_options
                -write_activity_file "shifted_activity.fsdb"
pwr_shell> update_power
```

The following example rejects glitch pulses whose width is less than 75% of the cell delay.

```
pwr_shell> set_power_delay_shifted_event_analysis_options -glitch_pulse_r
 75
```

Feedback

See Also

- power_enable_delay_shifted_event_analysis
- report_power_delay_shifted_event_analysis_options
- reset_power_delay_shifted_event_analysis_options

---

set_power_derate

Sets power derating factors for either the current design or a specified list of instances (cells, library cells, power groups, pins or nets).

Syntax

```
int set_power_derate
[-switching]
[-internal]
[-leakage]
[-x_transition]
[-rails rail_list]
[-pg_pin pg_pin_list]
[-groups group_name_list]
[-pin_switching pin_list]
derate_value
object_list
```

Data Types

group_name_list	list
rail_list	list
pg_pin_list	list
derate_value	float
object_list	list

Arguments

`-switching`

Indicates that the *derate_value* specified should be applied to switching power.

`-internal`

Indicates that the *derate_value* specified should be applied to internal power.

`-leakage`

Indicates that the *derate_value* specified should be applied to leakage power only. For CCS power library, it should be applied to both gate leakage and subthreshold leakage power.

`-x_transition`

Indicates that the *derate_value* specified should be applied to x-transition power. This feature is only valid for pins and nets objects.

`-groups` *`group_name_list`*

Specifies the power groups to which the *derate_value* is applied. The *derate_value* applies directly to each cell object in the power group.

`-rails` *`rail_list`*

Specifies a list of power supply nets on which power derate factor applied. This option is not supported for non-UPF mode. This feature is only valid on the following objects hierarchycal_cell, leaf_cell, Design and, power- groups. Derated values can be reported by *report_power_derate* command.

`-pg_pins` *`pg_pin_list`*

Specifies a list of pg_pins on which power derate factor applied. This feature is only valid on the following objects leaf_cell,and lib_cell. Derated values can be reported by *report_power_derate* command.

`-pin_switching`

Indicates that the *derate_value* specified should be applied to pin switching power. This feature is only valid for pins objects when variable *power_enable_pin_power_to_receiver_mode set to true.*

`derate_value`

Specifies the power derating factor that is applied to the specified power components as a scalar multiplicative factor.

`object_list`

Specifies current design or a list of cells or library cells to which the specified power derating factor is applied.

Description

Sets power derating factors on different power components for either the current design, a list of cells, library cells , pins or nets in the current design. If no object or power group is specified, the power derating factor (except for x-transition power derating factor) is set on the current design The power derating factors are used as a scalar multiplicative factor in power calculation.

Power derating factors affect power analysis results in power reports and power attributes. Power analysis results are multiplied by the derating factors. If power derating factors are not specified, the value of 1.0 is assumed (and value of 0.5 for x-transition power derating factor is assumed).

If you use the *set_power_derate* command with the *-internal*, *-switching*, or *-leakage* option, the specified power derating factor is only be applied to internal, switching, or leakage power accordingly. If you do not specify the *-internal*, *-switching*, or *-leakage* option, the power derating factor applies to all power components, which includes internal, switching, and leakage power. You can first set a generic power derating factor that applies to all power components, and then refine it by setting specific power derating factors for particular power components on particular design objects.

The *object_list* option can be used to set power specific derating factors on instances (cells, library cells, pins or nets) in the design. On each instance the *-switching*, *-internal*, and *-leakage* options can be used in the same way as previously described to specify exactly how the derating factors should be applied to each instance in the object list.

When applying power derating factors the following priority is used (in decreasing order of precedence):

1) Leaf Cell or Power Group

2) Hierarchical Cell

3) Library Cell

4) Design

Therefore, when power derating factors are being applied for a cell, PrimePower first checks if they have been set for the cell itself (leaf cell). It then checks if it has been set for any of the cells hierarchical parents. If no derate factors are found, it checks for the library cell and after that it uses the factors set globally on the design.

If power derating factors are set multiple times on the same design object for the same power component, the later value overrides the previous value.

Use the *-groups* option to set specific derating factors on all the cell objects in particular power groups. Power group can be user generated or default, such as clock_network, memory, register. They are considered as a collection of leaf and hierarchical cells. When applying power derating factors, the factors are set directly on the cell objects in the collection. There is one exception. Currently in PrimePower, clock network power includes the clock pin power of registers by default. The clock network power derating factors are not applied to registers, but only applied to the clock pin power of the registers when it was included in the clock network power. The power derating factors set on clock network power group are reset with the *reset_power_derate* command on current design or without any option.

In addition, if the *-switching* option is specified and the *object_list* contains any hierarchical cells, all cells within that hierarchical cell and also all lower hierarchical cells have that derating factor set on them.

The *-groups* option can be used to set specific derating factors on all the cell objects in particular power groups. Power group can be user generated or default, such as

clock_network, memory, register, etc.. If a derate factor is set on a power group, the specified derating factors are applied to all cell objects in the power group.

Power derating factors affect power values shown in power reports. The intended usage mode of this command is that you first set global or default derates on the design. Then you can use the *object_list* option to set specific derate values for cells or library cells in the design. To set derates globally on a design simply issue the command with no object list specified.

To set derating values back to the default (derating factor of 1.0 and x-transition derating factor of 0.5 for every instance in the design), use the *reset_power_derate* command.

Examples

The following example sets a power derating factor of value 0.95 on all cells in the design.

```
pt_shell> set_power_derate 0.95
```

The following example sets switching power derate and internal power derate to 0.9.

```
pt_shell> set_power_derate 0.9 -switching -internal
```

The following example sets x-transition power derate to 0.75 on pin U0/CP.

```
pt_shell> set_power_derate 0.75 -x_transition [get_pin U0/CP]
```

The following example sets the power derate to 0.5 for all power components on the default clock network power group.

```
pt_shell> set_power_derate 0.5 -groups clock_network
```

The following example sets a leakage power derating factor of 1.1 on all instances of library cell IV in the library MY_LIB:

```
pt_shell> set_power_derate -leakage 1.1 [get_lib_cells MY_LIB/IV]
```

Assuming that cell, "inv" is a particular instantiation of MY_LIB/IV in the design the following command overwrites the power derating factor for the instance "inv" only:

```
pt_shell> set_power_derate -leakage 1.05 [get_cells inv]
```

The following example illustrates how to set a power derating factor except leakage power on all cells (including sub-hierarchies) within the hierarchy top/H1:

```
pt_shell> set_power_derate -switching -internal 1.05 [get_cells top/H1]
```

```
pt_shell> set_power_derate -pin_switching 1.05 [get_pins
 add_1_root_add_27_2/U1_0/A]
```

Feedback

See Also

- report_power
- report_power_calculation
- report_power_derate
- reset_power_derate

---

set_power_profile_options

Sets the options for power profile generation.

Syntax

status *set_power_profile_options*

```
[-enable_time_based true | false]
[-interval nterval]
[-start_time event_start_time]
[-end_time end_time]
[-row row]
[-column column]
[-grid_size_x grid_size_x]
[-grid_size_y grid_size_y]
[-physical_hierarchy object_list]
[-count_toggle_by_total_power true | false]
```

Data Types

interval	float
start_time	float
end_time	float
row	integer
column	integer
grid_size_x	float
grid_size_y	float
physical_hierarchy	list

Arguments

`-enable_time_based`

Enables generating time based power profile. Default is false, which means a static power profile is generated. When this option is set to true, an Excel spreadsheet of format XLSX is used to generate time based power profile, where each sheet represents power profile from each time interval.

`-interval`

Specifies time interval for generation of time based power profile. Default interval is the dominant clock period of the current design.

`-start_time`

Specifies start time for time based power profile. Default is the start time of the FSDB file from which power profile is generated.

`-end_time`

Specifies end time for time based power profile. Default is the end time of the FSDB file from which power profile is generated.

`-row`

Specifies number of rows in the power profile. When the option *-physical_hierarchy* is not specified, the row specification is applied to the whole design. When the option *-physical_hierarchy* is specified, the row specification is applied to that physical hierarchy. This option cannot be use with the option *-grid_size_y*.

`-column`

Specifies number of columns in the power profile. When the option *-physical_hierarchy* is not specified, the column specification is applied to the whole design. When the option *-physical_hierarchy* is specified, the column specification is applied to that physical hierarchy. This option cannot be use with the option *-grid_size_x*.

`-grid_size_x`

Specifies grid size in X direction for power profile. When the option *-physical_hierarchy* is not specified, the grid size specification is applied to the whole design. When the option *-physical_hierarchy* is specified, the grid size specification is applied to that physical hierarchy. This option cannot be use with the option *-column*.

`-grid_size_y`

Specifies grid size in Y direction for power profile. When the option *-physical_hierarchy* is not specified, the grid size specification is applied to the whole design. When the option *-physical_hierarchy* is specified, the grid size specification is applied to that physical hierarchy. This option cannot be use with the option *-row*.

`-physical_hierarchy`

Specified a list of physical hierarchial cell objects such that the profile specification such as number of rows, columns, and grid size in X/Y direction is

applied to. When this option is omitted, the profile specification is applied to the whole design.

`-count_toggle_by_total_power`

Specifies whether to use total power with a predefined threshold to determine if a toggle occurs in the FSDB file. Default is true. When this option is set to false, toggle will be counted by value changes on internal power and switching power in the FSDB file.

Description

This command is used to specify options for the command *generate_power_profile*.

For power profile generation, user need to provide LEF/DEF files through the command *set_eco_options* with options *-physical_tech_lib_path -physical_lib_path -physical_design_path*.

For static power profile generation, static power analysis has be run first before invoking the command *generate_power_profile*.

Examples

The following example sets number of rows as 20, and number of columns as 10 for power profile generation.

```
pt_shell> update_power
pt_shell> set_eco_options -physical_tech_lib_path tech.lef
pt_shell> set_eco_options -physical_lib_path design.lef
pt_shell> set_eco_options -physical_design_path design.def
pt_shell> set_power_profile_options -row 20 -column 10
pt_shell> generate_power_profile power.csv
```

The following example enables time based power profile generation.

```
pt_shell> set_power_profile_options -enable_time_based true
pt_shell> generate_power_profile power.xlsx
```

The following example specifies different number of rows and columns between top level design and a physical block A.

```
pt_shell> set_power_profile_options -row 10 -column 10
 -physical_hierarchy A
pt_shell> set_power_profile_options -row 20 -column 20
pt_shell> generate_power_profile physical.csv
```

See Also

- generate_power_profile

Feedback

set_power_scenario

Specify scenario name for power analysis.

Syntax

int *set_power_scenario*

```
        scenario_name
```

Data Types

scenario_name	string

Arguments

scenario_name

Specifies scenario for power analysis. PrimePower RTL will use operating condition of given scenario to choose proper reference libraries.

Description

The *set_power_scenario* command specifies scenario for power analysis purpose. For scenario for synthesis, it follows defined scenario in synth option setting.

Examples

The following example shows how to specify different scenario for power. In synth options, two scenarios are defined 'func.fast' and 'func.slow'. Current scenario in synth option is 'func.fast' but user want to use 'func.slow' for power analysis.

```
pwr_shell> set_power_scenario func.slow
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_phy_tech_file
- set_synth_output_dir
- set_rtl_architect_shell
- set_synth_options
- set_postsynth_options

Feedback

set_pprtl_flow_options

Specify PrimePower RTL flow options

Syntax

int *set_pprtl_flow_options*

```
[-ignore_version_checks]
[-enable_variable_restore]
[-disable_hard_errors]
[-reset]
```

Arguments

-ignore_version_checks

Specify this option to skip tool version compatibility checks when an existing workspace is reused.

-enable_variable_restore

Specify this option to restore the previous state variables when reusing a workspace. This option creates and saves the state of all app vars and user variables into the workspace in a separate file. When the workspace is reused, the tool restores the state variables files automatically.

-disable_hard_errors

This option is used to prevent the tool RTL from exiting on errors.

-reset

This option is used to clear all previous settings of the set_pprtl_flow_options command.

Description

The *set_pprtl_flow_options* command sets the options to configure the PrimePower RTL flow.

Examples

The following example automatically saves and restores a PrimePower RTL session.

```
pwr_shell> set_pprtl_flow_options -enable_variable_restore
```

```
The following example restores the session and ignore version checks.
```

```
pwr_shell> set_pprtl_flow_options -enable_variable_restore
 -ignore_version_checks
```

Feedback

```
The following example restores the session, ignore version checks and
 disables hard errors.

pwr_shell> set_pprtl_flow_options -enable_variable_restore
 -ignore_version_checks -disable_hard_errors

The following example clears all previous settings.

pwr_shell> set_pprtl_flow_options -reset
```

See Also

- compute_metrics

---

set_prepower_options

Specify PrimePower RTL commands to be executed before update_power

Syntax

int *set_prepower_options*

```
[-file file_name]
[-cmds cmd_list]
```

Data Types

```
file_name       string
cmd_list        list
```

Arguments

`-file file_name`

Specifies tcl file which contains tcl commands to be executed before update_power.

`-cmds cmd_list`

if number of tcl commands is small, specify commands directly instead of creating separate file.

Description

The *set_prepower_options* command pass PrimePower RTL tcl commands to be executed before update_timing/update_power. To set options for timing/power use this command.

Examples

The following example shows how to pass arbitrary command

Feedback

```
pwr_shell> set_prepower_options -cmds {
              set_propagated_net [get_net CLK]
           }
```

See Also

- compute_metrics
- set_rtl_activity_file

---

set_presynth_options

Specify RTL Architect commands to be executed before analyze/elaborate.

Syntax

int *set_presynth_options*

```
[-file file_name]
[-cmds cmd_list]
```

Data Types

```
file_name       string
cmd_list        list
```

Arguments

`-file file_name`

Specifies tcl file which contains tcl commands to be executed before analyze/elaborate.

`-cmds cmd_list`

if number of tcl commands is small, specify commands directly instead of creating separate file.

Description

The *set_presynth_options* command pass RTL Architect tcl commands to be executed before analyze/elaborate. To set options for analyze/elaborate, use this command.

Examples

The following example shows how to pass arbitrary command

```
pwr_shell> set_presynth_options -cmds {
              puts "*** before analyze ***"
           }
```

Feedback

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_phy_tech_file
- set_synth_output_dir
- set_rtl_architect_shell
- set_synth_options
- set_postsynth_options

---

set_program_options

Sets options that affect the performance and capacity.

Syntax

status *set_program_options*

```
[-enable_high_capacity]
[-disable_high_capacity]
```

Arguments

`-enable_high_capacity`

Runs the tool in high capacity mode, which reduces the peak memory footprint.

`-disable_high_capacity`

Disables the high capacity mode of the program and returns to the default normal analysis mode, which usually increases the peak memory footprint.

Description

This command provides options that trade off performance and capacity without affecting the quality of the timing analysis results.

High Capacity Mode

In high capacity mode, the tool temporarily stores data to a local disk partition specified by the *pt_tmp_dir* variable; the default is the /tmp directory. When you exit the session, the consumed disk space is automatically released. For effective use of this mode, it is recommended that the capacity of the local disk on the host machine should be as large

Feedback

as the physical RAM. The potential capacity improvement might not be fully realizable in case of insufficient disk space.

Using high capacity mode does not change the timing analysis results. You can enable or disable high capacity mode only when there are no designs and libraries loaded in the program memory.

By default, high capacity mode is enabled. To see whether high capacity mode is enabled, check the setting of the *sh_high_capacity_enabled* read-only variable.

To specify the actual capacity improvement in high capacity mode, set the *sh_high_capacity_effort* variable.

Examples

The following command enables high capacity mode with the default settings:

```
pt_shell> set sh_high_capacity_effort high
Information: The setting will be effective after next
 'set_program_options' command. (PTHC-001)
high

pt_shell> set_program_options -enable_high_capacity
Information: enabling high capacity analysis in 'high' effort........
 (PTHC-001)
pt_shell> printvar sh_high_capacity_enabled
sh_high_capacity_enabled = "true"
```

See Also

- pt_tmp_dir
- sh_high_capacity_effort
- sh_high_capacity_enabled

---

set_propagated_clock

Specifies propagated clock latency.

Syntax

string *set_propagated_clock*

object_list

Data Types

object_list list

Feedback

Arguments

object_list

Specifies a list of clocks, ports, or pins.

Description

Specifies that delays be propagated through the clock network to determine latency at register clock pins. If it is not specified, ideal clocking is assumed. Ideal clocking means clock networks have a specified latency (designated by the *set_clock_latency* command), or zero latency, by default. Propagated clock latency is used for post-layout, after final clock tree generation. Ideal clock latency provides an estimate of the clock tree for pre-layout.

If the *set_propagated_clock* command is applied to pins or ports, it affects all register clock pins in the transitive fanout of the pins or ports. If it is applied to an object (clock, port, or pin) on which network latency is already defined, then a warning is issued.

Virtual clocks do not have any source, and they cannot affect any register in the design. Thus, virtual clocks cannot be made propagated, and an error is issued if the *set_propagated_clock* command is applied on a virtual clock.

To undo the *set_propagated_clock* command, use either the *remove_propagated_clock* or *set_clock_latency* command to provide an ideal latency.

To see propagated clock attributes on clocks, use the *report_clock* command.

Limitations: CRPR does not support propagated clocks set on pins or ports unless they are source pins or ports of common clock for a path. If the *timing_remove_clock_reconvergence_pessimism* variable is set to TRUE, then propagated clocks should be defined on clock objects, not pins or ports.

Examples

The following example specifies using propagated clock latency for all clocks in the design.

```
pt_shell> set_propagated_clock [all_clocks]
```

See Also

- timing_remove_clock_reconvergence_pessimism
- remove_propagated_clock
- report_clock
- set_clock_latency

Feedback

set_pulse_clock_max_transition

Sets the maximum transition for pulse generator input and pulse clock network with respect to the main library trip-points.

Syntax

string *set_pulse_clock_max_transition*

```
transition_value
[-rise]
[-fall]
[-transitive_fanout]
object_list
```

Data Types

transition_value	float
object_list	list

Arguments

`-rise`

Specifies rising transition for all selected pins.

`-fall`

Specifies falling transition for all selected pins.

`-transitive_fanout`

Specifies the constraint is set at the transitive fanout of pulse generator. If this option is not set, only the input of the pulse generators are constrained.

transition_value

Sets the transition limit (Value >= 0). This is the maximum transition time in main library time units.

object_list

Provides a list of pulse generator cell, pulse generator lib cell, clock and design.

Description

Specifies the maximum transition in the pulse clock network at the forward of pulse generator cell, pulse generator lib cell, clock and design if *-transitive_fanout* option is set. By default, the maximum transition constraint is set at the input of the pulse generator of the specified object list. If the cell or lib cell is not a pulse generator, or if the clock or design does not have any pulse generators, then this maximum transition limit has no effect. The pulse clock network or the input of pulse generator is expected to have transition time less than the specified the *transition_value* option.

The maximum transition constraint set on design by the *set_max_transition* command applies to the pulse clock network and input of pulse generator as also maximum transition constraint set on the pins in the library. Transition constraints are interpreted in main library trip-points and derate. During slack computation, the constraints are scaled to the pin trip-points and derate. In the case of conflicting constraints on a pin, the tightest constraint applies.

The maximum transition constraint can be optionally restricted to rising transitions only or falling transitions only by using the *-rise* or *-fall* options respectively. If the *-rise* or *-fall* options are not specified, both rise and falling transitions are constrained.

The *report_constraint -pulse_clock_max_transition* and *report_pulse_clock_max_transition* commands show the maximum transition constraint evaluations. The limit with appropriate scaling used for slack computation is reported.

To remove maximum pulse clock transition limits, use the *remove_pulse_clock_max_transition* command.

Examples

The following example sets a maximum transition limit of 0.4 units on input of all the cells corresponding to lib cell pulse_rise_high in library celllib.

```
pt_shell> set_pulse_clock_max_transition 0.4 {"celllib/pulse_rise_high"}
```

The following example sets a maximum transition limit of 0.4 units on all the pulse clock networks in the clock network clk.

```
pt_shell> set_pulse_clock_max_transition -transitive_fanout 0.4
 [get_clocks clk]
```

See Also

- current_design
- remove_pulse_clock_max_transition
- report_pulse_clock_max_transition
- report_constraint

---

set_pulse_clock_max_width

Sets the maximum pulse width constraint for pulse generator network.

Syntax

string *set_pulse_clock_max_width*

```
[-transitive_fanout]
pulse_width_value
object_list
```

Data Types

```
pulse_width_value       float
object_list             list
```

Arguments

`-transitive_fanout`

The constraints are applied to the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

`pulse_width_value`

Sets the pulse width limit (Value >= 0).

`object_list`

Provides a list of pulse generator cell, pulse generator lib cell, clock and design.

Description

Specifies the maximum pulse width limit for the pulse clock network at the forward of pulse generator cell by pulse generator instance name or pulse generator library cell. Setting the pulse width constraint by clock applies the constraint to the forward of all pulse generators driven by the clock. Setting by pulse width constraint by design applies the constraint to all the pulse clock networks in the design. In the presence of a conflicting constraint, the most restrictive constraint is used. If the cell or lib cell is not a pulse generator, or clock or design does not have any pulse generators, then this maximum width limit has no effect. The pulse clock network is expected to have pulse width time less than the specified *pulse_width_value* option.

The high or low pulse width constraint is inferred based on sense propagation. For example, after an inverter, a high pulse width constraint is converted to a low pulse width constraint.

The *report_constraint -pulse_clock_max_width* and *report_pulse_clock_max_width* commands show maximum pulse width constraint evaluations.

To remove maximum pulse width limits, use the *remove_pulse_clock_max_width* command.

Examples

The following example sets a maximum pulse width limit of 0.4 units on all the pulse clock networks in the design.

```
pt_shell> set_pulse_clock_max_width 0.4 [current_design]
```

Feedback

See Also

- current_design
- remove_pulse_clock_max_width
- report_pulse_clock_max_width
- report_constraint

---

set_pulse_clock_min_transition

Sets the minimum transition at the input of pulse generator with respect to the main library trip-points.

Syntax

string *set_pulse_clock_min_transition*

```
transition_value
[-rise]
[-fall]
object_list
```

Data Types

transition_value	float
object_list	list

Arguments

`-rise`

Specifies rising transition for all selected pins.

`-fall`

Specifies falling transition for all selected pins.

transition_value

Sets the transition limit (Value >= 0). This is the minimum transition time in main library time units.

object_list

Provides a list of pulse generator cell, pulse generator lib cell, clock and design.

Description

Specifies the minimum transition at the input of pulse generator cell and pulse generator lib cell. If the constraint is specified on the clock and design, all the inputs of pulse generators in the clock network and design respectively are constrained. If the cell or lib

cell is not a pulse generator or clock or design does not have any pulse generators, then minimum pulse clock transition limit has no effect. The input of pulse generator is expected to have transition time greater than the specified *transition_value* option.

Transition constraints are interpreted in main library trip-points and derate. During slack computation, the constraints are scaled to the pin trip-points and derate. In the case of conflicting constraints on a pin, the tightest constraint applies.

The minimum transition constraint can be optionally restricted to rising transitions only or falling transitions only by using the *-rise* or *-fall* options, respectively. If the *-rise* or *-fall* options are not specified, both rise and falling transitions are constrained.

The *report_constraint -pulse_clock_min_transition* and *report_pulse_clock_min_transition* commands show minimum transition constraint evaluations. The constraint with appropriate scaling used for slack computation is reported.

To remove minimum pulse clock transition limits, use the *remove_pulse_clock_min_transition* command.

Examples

The following example sets a minimum transition limit of 0.4 units on input of all the cells corresponding to lib cell pulse_rise_high in library celllib.

```
pt_shell> set_pulse_clock_min_transition 0.4 {"celllib/pulse_rise_high"}
```

See Also

- current_design
- remove_pulse_clock_min_transition
- report_pulse_clock_min_transition
- report_constraint

---

set_pulse_clock_min_width

Sets minimum pulse width constraint for pulse generator network.

Syntax

string *set_pulse_clock_min_width*

```
pulse_width_value
[-transitive_fanout]
object_list
```

Feedback

Data Types

pulse_width_value	float
object_list	list

Arguments

`-transitive_fanout`

The constraints are applied to the transitive fanout of pulse generators. In this release, specifying or not specifying this option has the same behavior.

pulse_width_value

Sets the pulse width limit (Value >= 0).

object_list

Provides a list of pulse generator cell, pulse generator lib cell, clock and design.

Description

Specifies the minimum pulse width limit for the pulse clock network at the forward of pulse generator cell by pulse generator instance name or pulse generator library cell. Setting the pulse width constraint by clock applies the constraint to the forward of all pulse generators driven by the clock. Setting by pulse width constraint by design applies the constraint to all the pulse clock networks in the design. Minimum pulse width constraint from the library also applies to the pulse clock network. In the presence of a conflicting constraint, the most restrictive constraint is used. If the cell or lib cell is not a pulse generator or clock or design does not have any pulse generators, then this minimum width limit has no effect. The pulse clock network at the forward of pulse generator is expected to have width more than the specified *pulse_width_value* option.

The high or low pulse width constraint is inferred based on sense propagation. For example, after an inverter, a high pulse width constraint is a low pulse width constraint.

The *report_constraint -pulse_clock_min_width* and *report_pulse_clock_min_width* commands show minimum pulse width constraint evaluations.

To remove minimum pulse width limits, use the *remove_pulse_clock_min_width* command.

Examples

The following example sets a minimum pulse width limit of 0.4 units on all the pulse clock networks in the clock network clk:

```
pt_shell> set_pulse_clock_min_width 0.4 [get_clocks clk]
```

Feedback

See Also

- current_design
- remove_pulse_clock_min_width
- report_pulse_clock_min_width
- report_constraint

---

set_pvt_explorer_condition

Sets PVT explorer condition for PrimeShield PVT Variation Exploration what-if analysis.

Syntax

string *set_pvt_explorer_condition*

```
        input_file
```

Data Types

input_file string

Arguments

input_file

Specifies the PVT Explorer condition input file.

Description

The *set_pvt_explorer_condition* command set PVT explorer condition, which is used in PrimeShield Power, voltage, and temperature (PVT) parameter variation and exploration.

define_sensitivity_lib_mapping is required in the feature. And *set ps_enable_pvt_explorer_analysis true* is required to use *set_pvt_explorer_condition* command.

Examples

The following script enables Power, performance, area (PPA) what-if analysis, which enables you to change the design PVT parameters and quickly assess their PPA impact. It allows process sweet-spot exploration and technology option assessment.

"one_cond.txt" file content as follows:
#index,pmos_lvt_vt,nmos_lvt_vt,pmos_lvt_ids0,nmos_lvt_ids0 1,0.020,0.020,0.15,0.15

The timing and power analysis results after *set_pvt_explorer_condition* will reflect the physical parameter shift defined in "one_cond.txt".

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_pvt_explorer_analysis true
set ps_pvt_explorer_pba_only_mode false
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]

set_pvt_explorer_condition one_cond.txt
update_timing -full
update_power
```

You must change the "one_cond.txt" file content to include the device model names and values, based on your PDK or side file. The "one_cond.txt" file includes the delta shift values. In the preceding example, the vt shift provided is +20mV and the idsat shift provided is +15%, from their respective nominal values. For SPICE correlation, the actual value, that is, the nominal value plus the delta value, must be specified with the *-sim_option* option. In the above example, it will be like following: *sim_setup_simulator -sim_option ".param pmos_lvt_vt=0.020 nmos_lvt_vt=0.020 pmos_lvt_ids0=1.15 nmos_lvt_ids0=1.15"*

Timing robustness what-if analysis enables you to perform path-based timing analysis with multiple what-if specifications. It allows you to identify paths susceptible to PVT changes, o provide ECO guidance to improve the design robustness.
The following script enables what-if analysis for timing path robustness, where "multiple_DoEs.txt" file content as follows to cover multiple what-if specifications:
#index,pmos_lvt_vt,nmos_lvt_vt,pmos_lvt_ids0,nmos_lvt_ids0 1,0.020,0.020,0.15,0.15 2,0.011,0.011,0.10,0.10

```
set link_path orig.db
read_verilog ...
link_design
set ps_enable_analysis true
set ps_enable_pvt_explorer_analysis true
set ps_pvt_explorer_pba_only_mode true
define_sensitivity_lib_mapping -side_file side.db [get_lib orig]
update_timing -full  # baseline QoR
set paths [get_timing_paths ...]

set_pvt_explorer_condition multiple_DoEs.txt
#update original path collection with slack shift per DoE
set paths_new [get_timing_paths -pba_mode path $paths]
#generate slack shift per DoE for all paths
puts [get_attr $paths_mew pvt_explorer_slack_shift]
```

The slack of paths in paths_new collection will be the worst among all what-if specifications, and attribute *pvt_explorer_slack_shift* is a list of float for per specification slack shift.

Feedback

See Also

- define_sensitivity_lib_mapping
- report_pvt_explorer_condition
- reset_pvt_explorer_condition
- ps_enable_spice2design_analysis
- ps_enable_pvt_explorer_analysis
- ps_pvt_explorer_pba_only_mode

---

set_qtm_attribute

Sets an attribute to the specified value on QTM object(s).

Syntax

string *set_qtm_attribute*

```
-class class_name
attr_name
value
[object_names]
```

Data Types

class_name	string
attr_name	string
value	string
object_names	list

Arguments

-class class_name

Specifies the class of the QTM objects to set attribute on. Allowed values are *lib*, *lib_cell*, or *lib_pin*.

attr_name

Specifies the name of the attribute.

value

Specifies the value of the attribute.

object_names

Specifies the names of the objects on which to set the attribute. Each element in the list is a name to identify an object in the specified class. The object names

should not be specified when the object class is either a *lib* or *lib_cell* value. It must be specified when the class is a *lib_pin* value, in which case the object names should be the names of ports already defined for the QTM.

Description

The *set_qtm_attribute* command sets attributes on objects. These attributes are either application attributes that are reserved by PrimeTime or defined with the *define_qtm_attribute* command as the user attributes. Please note the order of the command arguments. Also note that PrimeTime the *list_attribute*, *report_attribute*, and *get_attribute* commands do not work on attributes defined or set on the QTM objects.

Examples

The following example defines a Boolean attribute 'is_XYZ' for QTM cell, then sets the value on the QTM cell under construction.

```
pt_shell> define_qtm_attribute -type boolean -class lib_cell "is_XYZ"
pt_shell> set_qtm_attribute -class lib_cell "is_XYZ" "true"
```

The following example sets the attribute 'max_transition' for 2 QTM ports. Because 'max_transition' is an application reserved attribute for the object class 'lib_pin', it should be set directly.

```
pt_shell> set_qtm_attribute -class lib_pin "max_transition" 1.0 {IN, CLK}
```

See Also

- define_qtm_attribute
- remove_qtm_attribute
- define_user_attribute

---

set_qtm_global_parameter

Sets a global parameter for a quick timing model.

Syntax

string *set_qtm_global_parameter*

```
[-param setup | hold | recovery |removal | clk_to_output]
[-lib_cell lib_cell]
[-pin pin_name]
[-clock pin_name]
[-value parameter_value]
```

Data Types

```
lib_cell              string
pin_name              string
parameter_value       float
```

Arguments

`-param setup | hold | recovery |removal | clk_to_output`

Sets one of the following global parameters: *setup*, *hold*, *recovery*, *removal* or *clk_to_output*.

`-lib_cell` *lib_cell*

Specifies the library cell that you want to use to mimic the global parameter.

`-pin` *pin_name*

Specifies the related pin for the global parameter if you are using the *-lib_cell* option to mimic the behavior. If the parameter is *setup* or *hold*, this pin must be an input pin. If the parameter is *clk_to_output*, this pin must be an output pin. If you do not use this option, the quick timing model uses the first encountered input pin that is constrained, in the case of the *setup* or *hold* parameter; and the first output pin that has an edge delay arc coming into it, in the case of the *clk_to_output* parameter.

`-clock` *pin_name*

Specifies the constraining pin in the case of *setup* or *hold*; and the launch pin in the case of *clk_to_output* parameter.

`-value` *parameter_value*

The value of the global parameter in time units.

Description

This command sets the value of a quick timing model (QTM) global parameter. The global parameter can be either *setup*, *hold*, , *recovery*, *removal* or *clk_to_output*. These parameters specify the global parameter which is added to the arcs. To the QTM setup/hold/recovery/removal arc, the time is added to arrive at the final delay. To the QTM edge triggered delay arc the global clk_to_output time is added to arrive at the final delay.

You can specify the global parameters of the sequential element by specifying a cell in the library (the *-lib_cell* option), or by giving the actual value, by using the *-value* option. If the *-lib_cell* option is specified, you can specify the clock pin and the input pin name. If you do not specify one, the first setup, hold, or clk_to_output arc encountered is chosen (based on which parameter is being set). If only the *-clock* option is specified, the first setup, hold, or clk_to_output arc (depending on the global parameter) from that clock pin is chosen. If only the *-pin* option is specified, the first setup, hold, or clk_to_output arc (depending on the type of global parameter) coming into the pin is chosen.

Feedback

To report information about the current quick timing model, use the *report_qtm_model* command.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

In the following examples, before using the *-lib_cell* option, you must load the technology library and set the QTM technology. To do this, use the following sequence of commands:

```
read_db my_technology_library.db
set_qtm_technology -library my_technology_library
```

The following example sets the global setup time equivalent to the setup time of library cell DFF1:

```
pt_shell> set_qtm_global_parameter -param setup -lib_cell DFF1
```

The following example sets the global hold time equivalent to the setup time of pin D of library cell DFF1:

```
pt_shell> set_qtm_global_parameter -param setup -lib_cell DFF1 -pin D
```

The following example sets the global clk_to_out time to be 1.5 time units:

```
pt_shell> set_qtm_global_parameter -param clk_to_output -value 1.5
```

See Also

- create_qtm_constraint_arc
- create_qtm_delay_arc
- create_qtm_drive_type
- create_qtm_load_type
- create_qtm_model
- create_qtm_path_type
- report_qtm_model
- save_qtm_model

---

set_qtm_port_drive

Sets the drive on the quick timing model port.

Syntax

string *set_qtm_port_drive*

```
[-type drive_type]
[-value drive_value]
[-input_transition_rise rtrans]
[-input_transition_fall ftrans]
[-subtract_max_delay_from_total]
port_list
```

Data Types

drive_type	string
drive_value	float
rtrans	float
ftrans	float
port_list	list

Arguments

`-type` *drive_type*

Specifies the global *drive_type* parameter.

`-value` *drive_value*

Specifies the drive value in terms of drive resistance units.

port_list

Specifies a list of input or inout ports on which to set the attribute. Each element in the list is either a collection of quick timing model ports or a pattern that matches quick timing model ports.

`-input_transition_rise` *rtrans*

Specifies the input rising transition time associated with the *-input_pin* option of the specified drive type to compute the delay for the drive arc for this port. If you do not use this option, the delay table for rising input of the drive arc is loaded from library as is.

Use the *-input_transition_rise* and *-input_transition_fall* options to capture the transition time associated with the *-input_pin* option defined for the drive type referred. This can obtain more accurate information on the transition time and delay time for the drive arc defined for this port.

`-input_transition_fall` *ftrans*

Specifies the input falling transition time associated with the *-input_pin* option of the specified drive type to compute the delay for the drive arc for this port. If you do not use this option, the delay table for falling input of the drive arc is loaded from library as is.

Feedback

`-subtract_max_delay_from_total`

Indicates that the drive arc delay computed with max load is subtracted from the total delay budgeted to this output port. This affects all types of delay arcs that reach this output port. By default, the drive arc delay is added on top of the quick timing model delay arcs created to the output port and therefore increase the actual delay.

Description

This command sets the drive of the output port of the quick timing model. There are two methods for setting the drive:

- Use one of the global *drive_type* parameters, if you have drive types defined.
- Define the drive in terms of the drive resistance units.

When drive is defined for an output port, by default, the actual delay to the output port is the delay arc defined by the *create_qtm_delay_arc* command for the port plus the delay computed for this drive arc. If in your QTM creation, the delay defined by the *create_qtm_delay_arc* command for the port is actually the maximum delay budgeted for the output and you do not want the drive arc to add extra delay on top of that budget, you can optionally use the *-subtract_max_delay_from_total* option to keep the delay with maximum load at port unchanged. But with smaller load at the output port, the delay might actually be smaller than the budgeted maximum delay.

To define drive types, use the *create_qtm_drive_type* command.

To show the information about the current quick timing model, use the *report_qtm_model* command.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following command sets the drive of the OUT1 output port to be the drive1 drive type.

```
pt_shell> set_qtm_port_drive -type drive 1 OUT1
```

The following command sets the drive of the OUT2 output port to be 5 drive units.

```
pt_shell> set_qtm_port_drive -value 5.0 OUT2
```

See Also

- create_qtm_load_type
- create_qtm_model
- report_qtm_model
- save_qtm_model

set_qtm_port_load

Sets the load on quick timing model (QTM) ports.

Syntax

string *set_qtm_port_load*

```
[-type load_type]
[-factor multiplication_factor]
[-value load_value]
port_list
```

Data Types

load_type	string
multiplication_factor	float
load_value	float
port_list	list

Arguments

`-type load_type`

Specifies the global *load_type* parameter.

`-factor multiplication_factor`

Specifies the multiplication factor for the load type to set the load on the quick timing model port.

`-value load_value`

Specifies the load value in the capacitance units.

port_list

Specifies a list of input or inout ports on which to set the attribute. Each element in the list is either a collection of quick timing model ports or a pattern that matches quick timing model ports.

Feedback

Description

This command sets the load of the input port of the quick timing model. There are two methods for setting the load:

- Use the global *load_type* parameter along with a multiplication factor if you have load types defined.
- Define the load in terms of the capacitance units.

To define load types, use the *create_qtm_load_type* command.

To show the information about the current quick timing model, use the *report_qtm_model* command.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following command sets the load on IN1 to be 2 times the global *load_type* load1.

```
pt_shell> set_qtm_port_load -type load1 -factor 2 IN1
```

The following command sets the load on IN2 to be 4.5 capacitance units.

```
pt_shell> set_qtm_port_load -value 4.5 IN2
```

See Also

- create_qtm_load_type
- create_qtm_model
- get_qtm_ports
- report_qtm_model
- save_qtm_model
- set_qtm_port_drive

---

set_qtm_technology

Sets the quick timing model (QTM) technology variables.

Syntax

string *set_qtm_technology*

```
[-library name]
[-max_transition trans_value]
[-min_transition trans_value]
```

```
[-max_capacitance cap_value]
[-min_capacitance cap_value]
[-wire_load_model wlm_name]
[-operating_condition opcond_name]
[-process process_value]
[-voltage voltage_value]
[-temperature temperature_value]
```

Data Types

```
name              string
trans_value       float
cap_value         float
wlm_name          string
opcond_name       string
process_value     float
voltage_value     float
temperature_value   float
```

Arguments

`-library name`

Specifies the library name for using library elements to define global parameters.

`-max_transition trans_value`

Specifies the maximum transition value.

`-min_transition trans_value`

Specifies the minimum transition value.

`-max_capacitance cap_value`

Specifies the maximum capacitance value.

`-min_capacitance cap_value`

Specifies the minimum capacitance value.

`-wire_load_model wlm_name`

Specifies the wire load model used while calculating delays.

`-operating_condition opcond_name`

Specifies the default operating condition name for the quick timing model in a multicorner, multimode (MCMM) design. Must be specified together with *-process*, *-voltage*, and *-temperature*.

`-process process_value`

Specifies the process scaling factor for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the

following options, you must use all of them: *-operating_condition*, *-process*, *-voltage*, and *-temperature*. Allowed values are 0.0 through 100.0.

`-voltage` *`voltage_value`*

Specifies the voltage value, in Volts, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: *-operating_condition*, *-process*, *-voltage*, and *-temperature*.

`-temperature` *`temperature_value`*

Specifies the temperature value, in degrees Celsius, for the default operating condition of the quick timing model in an MCMM design. When you want to use any of the following options, you must use all of them: *-operating_condition*, *-process*, *-voltage*, and *-temperature*.

Description

This command sets the various quick timing model technology parameters. Use the *-library* option if you use library cells to define path_types, drive_types, load_types, or if you set the other global parameters in terms of library cells. Before setting the library, you must ensure that the library is loaded.

If you use the *-wire_load_model* option, the path_type delays are calculated using the information. You can use the *-operating_condition* option to specify the default operating condition name for the quick timing model in a multicorner, multimode (MCMM) design. It must be specified together with *-process*, *-voltage*, and *-temperature* options.

To show the information about the current quick timing model, use the *report_qtm_model* command.

For basic information about quick timing models, see the *create_qtm_model* man page.

Examples

The following example sets the library to my_lib, which is in the my_lib.db file.

```
pt_shell> read_db my_lib.db
pt_shell> set_qtm_technology -library my_lib
```

The following command sets the *-wire_load_model* option to wlm1.

```
pt_shell> set_qtm_technology -wire_load_model wlm1
```

See Also

- create_qtm_model
- report_qtm_model
- save_qtm_model

Feedback

set_query_rules

Defines rules for rule-based query in golden UPF mode. In non-golden UPF mode, this command is ignored.

Syntax

status *set_query_rules*

```
[-hierarchical_separators separator_list]
[-bus_name_notations bus_name_list]
[-class class_list]
[-regsub regsub_list]
[-regsub_cumulative]
[-wildcard]
[-suffix suffix_list]
[-verbose]
[-nocase]
[-reset]
```

Data Types

separator_list	list
bus_name_list	list
class_list	list
regsub_list	list
suffix_list	list

Arguments

`-hierarchical_separators` *separator_list*

Specifies a list of equivalent hierarchy separators in object names. There must be at least two elements in the list. The default list is {/ _ .}. The elements are preferably all single character, although at most one multicharacter element is allowed. No more than one alphabetical character, a-z or A-Z, is allowed as a hierarchical separator. The single alphabetical character element is also mutually exclusive with the multicharacter element. The following special characters are not supported as hierarchy separators nor they are allowed as one of the characters for the multicharacter hierarchy separator: 0-9, *, ?, \, +, ^, [,], (,), <, >, and {, }.

`-bus_name_notations` *bus_name_list*

Specifies a list of equivalent bus name notations in object names. There must be at least two elements in the list, and the list element must be of two characters exactly. The first character of the element is the opening bus name character, and the second character is the closing bus name character. The default list is {[] __ ()}. No more than one matching pair of alphabetical characters a-z or A-Z is allowed as bus name notations. The following characters are not supported as opening or closing bus name characters: 0-9, *, ?, \\, +, ^, and /. Bracketed

bus notations must be paired. For example, "[)" are not properly paired brackets, therefore it is not supported. For non-bracket bus name notations, the opening and closing bus name character must be the same. For example, "-_" is not a supported bus name notation.

`-class` *`class_list`*

Specifies a list of object class names for which the query rules are applied. Object classes cell, port, pin, net, power_domain, supply_net, supply_set, supply_port and power_switch are supported. The default is to enable all of them, however you could specify a subset of them.

`-regsub` *`regsub_list`*

Specifies a list of options for the *regsub* Tcl command. Each list should include three string elements: {{*options*} {*match_regexp*} {*substitution_exp*}}. Query uses the *-regsub* option to evaluate the object name and pattern strings during the matching process.

The *-regsub* option can be specified multiple times in one *set_query_rules* command. When multiple *-regsub* options are used, the *-regsub* options are applied to the leaf object name and pattern in the order you specify.

The *-regsub* options applies only to pin, port, net, and leaf cells. If *-regsub* is used without other query rules, the default values for the other options are still used. To save runtime, use this option only if the existing rule-based matching scheme fails to match an object.

In addition to the Tcl built-in options for the command *regsub*, the following options are supported:

- *-class* specifies a single object class name such as cell, port pin or net. The *-class* option makes the *-regsub* query rule object specific. Object class specific *-regsub* rule is the preferred to legacy global *-regsub* rule. The latter has been deprecated. For backward compatibility purposes, the global *-regsub* query rule is automatically converted to the class specific rules for port, pin and net, but not for cell class.
- *-cumulative* specifies that the multiple *-regsub* rules for the same object class are executed with cumulative effect in the sense that the subsequent *-regsub* rule is applied to the outputs of the object name and pattern string from the previous *-regsub* rules. After this option is used for an object class in one *-regsub*, all *-regsub* rules for that object class have the cumulative option.

`-regsub_cumulative`

Although this legacy option is supported, you should use *-regsub -cumulative* instead. If you use the *-regsub_cumulative* option, the tool automatically

converts to the class-specific *-cumulative* option for port, pin, and net, but not for cell class.

`-wildcard`

Enables wildcard support in rule-based match. Be cautious with this option because this option can end up matching more objects after ungroup and change_names.

`-suffix` *`suffix_list`*

Enables suffix name rule. Only the predefined suffix *_reg* and *_tri* are supported. Suffix *_reg* is only applicable to register cells. When enabled, a cell name pattern such as U4 would match the U4_reg register cell. Suffix *_tri* is only applicable to tristate cells. When enabled, a cell name pattern such as U8 would match the U8_tri tristate cell object.

`-verbose`

Prints messages indicating the object is matched using query rules.

`-nocase`

Enables case-insensitive rule-based match. This is the preferred way of enabling case-insensitive rule-based match.

`-reset`

Resets query rules to the program default.

Description

The *set_query_rules* defines query rules to guide the object queries in golden UPF mode. The purpose of the rule-based matching is to resolve the naming differences caused by altering names in Design Compiler and IC Compiler. The hierarchy separators and bus names are typical sources of the naming differences.

When a list of characters is defined as equivalent in terms of hierarchy separator, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list {/ _ .} is defined as equivalent hierarchy separators, the cell named "a.b_c/d_e" is matched with name string "a/b_c.d/e" provided in the golden UPF files.

When a list of bus notations is defined as equivalent in terms of bus names, the object query uses the equivalence when matching the objects in the in-memory netlist. For example, if the list {[] __} is defined as equivalent bus notations, the cell named "a[4][5]" is matched with the name string "a_4__5_" provided in the golden UPF files.

When the *-regsub* rules are specified, Tcl *regsub* command applies the rules to both the object name and the pattern string for rule-based matching if the other rule-based matching scheme fails to match an object.

Feedback

Be very cautious about enabling wildcard support in rule-based match, because the greedy nature of the wildcard match could often give you more matches than you want.

Use the *-reset* option to reset the query rules to the program default. Use *-show* to report the current query rules. When *-show* is used with other options, the new rules are set first and then reported.

The *-nocase* option of command *set_query_rules* is the preferred way of enabling case-insensitive rule-based matching, although global variable *find_ignore_case* variable is honored. For runtime reasons, avoid using the *-nocase* option in query commands such as *get_cells* or *get_pins*.

The *set_query_rules* command can be issued multiple times. Unless the *-show* option is used alone, the new rules always overwrite the previous rule settings.

Be cautious when defining query rules. To save runtime, do not use unnecessary query rules.

The *set_query_rules* command is effective only in golden UPF mode. Runtime for rule-based matching could be much slower.

Examples

```
# Show program default
prompt> set_query_rules -show

************************
Query Rules
************************

          Rule Type  Rule Value
------------------  --------------------
Hierarchy-Separator  {/ _ .}
       Bus-Notation  {[] __ ()}
       Object-Class  {cell port pin net power_domain power_switch
                      supply_net supply_set supply_port}
             Nocase  false
           Wildcard  false
            Verbose  false
1


# Non-default
prompt> set_query_rules \\
    -hierarchical_separators {/ _} \\
    -bus_name_notations {[] ||} \\
    -class {cell port} -show

************************
Query Rules
************************
```

```
            Rule Type   Rule Value
  ------------------   --------------------
  Hierarchy-Separator   {/ _}
         Bus-Notation   {[] ||}
         Object-Class   {cell port}
               Nocase   false
             Wildcard   false
              Verbose   false
1

# Use alphabetical characters
prompt> set_query_rules \\
    -hierarchical_separators {/ x} \\
    -class {cell port} \\
    -bus_name_notations {[] __ || xx} -show

************************
Query Rules
************************

            Rule Type   Rule Value
  ------------------   --------------------
  Hierarchy-Separator   {/ x}
         Bus-Notation   {[] __ || xx}
         Object-Class   {cell port}
               Nocase   false
             Wildcard   false
              Verbose   false
1

# Unsupported usage
prompt> set_query_rules \\
    -hierarchical_separators {/ x y} \\
    -bus_name_notations {[] __ || xx}
Error: At most 1 alphabetical or multicharacter hierarchy separator is
 allowed. (UID-1067)
0
```

See Also

- load_upf

---

set_rail_voltage

Sets power rail voltage on cells.

Syntax

int *set_rail_voltage*

```
[-rail_value rvalue]
[-rail_list rname_value_list]
[-min]
[-max]
[-dynamic_rail_value dynamic_component]
[-dynamic_rail_list name_value_list]
cell_list
```

Data Types

rvalue	float
rname_value_list	list
dynamic_component	float
name_value_list	list
cell_list	list

Arguments

`-rail_value` *rvalue*

Sets voltage on single-rail cells. If you use this option, you cannot use the *-rail_list* option because these options are mutually exclusive. Replace *rvalue* with a decimal or an integer value.

`-rail_list` *rname_value_list*

Sets voltages on individual rails of multirail cells. Replace *rname_value_list* with a list, which must have even number of elements. Odd elements are names of rails; even elements are voltage values.

The rail names must match the definitions of rails in the library power_supply section and in the library operating conditions. If you use this option, you cannot use the *-rail_value* option because these options are mutually exclusive.

`-min`

Sets rail voltages for the minimum operating condition. If you do not specify either the *-min* or *-max* option, the tool assumes you are using both the *-min* and *-max* options.

`-max`

Sets rail voltages for the maximum operating condition. If you do not specify either the *-min* or *-max* option, the tool assumes you are using both the *-min* and *-max* options.

`-dynamic_rail_value` *dynamic_component*

Specifies what portion of *rvalue* is dynamic. You can only specify this option if the *-rail_value* option is specified. If you attempt to specify this option with the *-rail_list* option, an error message is issued.

`-dynamic_rail_list` *`name_value_list`*

Specifies the dynamic portion of each rail voltage listed with the *-rail_list* option. The list of dynamic component values must match the entries in the *rname_value_list*.

`cell_list`

Specifies a list of cells on which to set rail voltages. Replace *cell_list* with the collection of cells you want.

Description

Sets power rail voltage on cells. This command overrides voltages specified in operating conditions. It can be applied to both leaf cells and hierarchical blocks. If used on hierarchical cells, all cells in the block must come from the same library, unless another *set_rail_voltage* or *set_operating_conditions* command has been set on them.

It should be noted that since *-min* and *-max* refer to operating conditions, the *-max* voltage is typically lower than the *-min* voltage.

The voltages are passed to delay equations in the library. Ground rail voltages cannot be changed; the tool assumes zero voltage.

In a typical multi-voltage flow you can set operating conditions on hierarchical blocks corresponding to voltage islands. You can then use a power network analysis tool to calculate the IR drop in power networks; subtract power and ground voltages on cells; and, by using the *set_rail_voltage* command, annotate them on important leaf cells or blocks.

The *-dynamic_rail_value* or *-dynamic_rail_list* option can be used to specify the dynamic component of IR drop. In this case, the *-rail_value* or *-rail_list* option can be used to specify the total voltage on a cell and the dynamic options can specify what portion of that voltage is dynamic.

Care should be taken when specifying the dynamic component of the rail voltage for the max operating condition. In this case, the dynamic component should be negative. This ensures that the static component of rail voltage is not less than the total minimum rail voltage (corresponding to max operating condition).

If Clock Reconvergence Pessimism Removal (CRPR) is enabled and dynamic rail voltages are specified, the Dynamic CRPR mode is turned on. In this mode only CRPR is computed using clock totals considering the dynamic component of rail voltage if the path is zero-cycle. A zero-cycle path is one in which the same clock edge both launches and captures.

Note that the operating condition (or rail voltage) that is set on the lower levels of the hierarchy (or leaf cell) overrides the operating condition (or rail voltage) on the higher levels of hierarchy.

Feedback

Examples

The following example sets the operating conditions named *OC1* and *OC2* on the block named *h1* and then overrides the voltage on the cell named *u3* in *h1*.

```
pt_shell> set_operating_conditions -min OC1 -max OC2 \\
     -object_list [get_cells {h1}]
1

pt_shell> set_rail_voltage -min -rail_value 1.4 [get_cells {h1/u3}]
1

pt_shell> set_rail_voltage -max -rail_value 1.1 [get_cells {h1/u3}]
1
```

To specify a dynamic component of 0.2 for the example above, you should use the following commands (Note the negative value for the max operating condition):

```
pt_shell> set_rail_voltage -min -rail_value 1.4 -dynamic_rail_value 0.2
 [get_cells {h1/u3}]
1

pt_shell> set_rail_voltage -max -rail_value 1.1 -dynamic_rail_value -0.2
 [get_cells {h1/u3}]
1
```

See Also

- remove_rail_voltage
- set_operating_conditions

---

set_related_supply_net

Associates an external supply net to the port of the design.

Syntax

status *set_related_supply_net*

```
[-object_list ports]
[-ground ground_net]
[-power power_net]
[-reset]
[power_net]
```

Data Types

ports	list
ground_net	string
power_net	string

Feedback

Arguments

`-object_list` *`ports`*

Specifies the list of ports (pins) to be associated with power/ground nets or that must be reset. If this option is not specified, all the ports are considered.

`-ground` *`ground_net`*

Specifies the ground net.

`-power` *`power_net`*

Specifies the power net.

`-reset`

Resets the related power and ground nets of the ports specified with the *-object_list* option. This cannot be used in conjunction with either the *-power* or *-ground* option.

`power_net`

Specifies the power net. This is deprecated and supported for backward compatibility only. New scripts should specify the power net using the *-power* option.

Description

This command is used to associate supply nets with design ports. The tool uses this information for constraining and checking design. The level shifter insertion tool also uses supply nets to determine if level shifter is required between a port and the logic connected to the port.

Setting related supply nets on pins is used for level shifter insertion in synthesis and design optimization tools. PrimeTime accepts pins in *-object_list* but does not use such information for its analysis.

If this command is not specified, the tool assumes that ports are externally connected to the primary supply net of the domain of the top design.

Examples

The following example sets the *VDD_EXT* power net and *VSS_EXT* ground net with *INPUT* port.

```
pt_shell> set_related_supply_net -object_list \\
[get_ports INPUT] -power VDD_EXT -ground VSS_EXT
```

This example resets the power net and ground net on all ports.

```
pt_shell> set_related_supply_net -reset
```

Feedback

See Also

- create_supply_net
- get_supply_nets
- get_ports

---

set_resistance

Sets the *ba_net_resistance* attribute with a resistance value on specified nets.

Syntax

int *set_resistance*

```
[-min]
[-max]
resistance_value
object_list
```

Data Types

```
resistance_value     float
object_list          list
```

Arguments

`-min`

Applies only for designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* option is the minimum resistance.

`-max`

Applies only for designs in min-max mode (min and max operating conditions). Indicates that the *resistance_value* option is the maximum resistance.

`resistance_value`

Specifies the resistance value for nets in the *object_list* option in units consistent with those used by the technology library.

`object_list`

A list of nets on which the *resistance_value* option is set.

Description

Sets the *ba_net_resistance* attribute, which specifies the back-annotation of resistance values on nets in the current design. If the current design is hierarchical, you must link it using the *link* command. The specified *resistance_value* option overrides the

Feedback

internally estimated net resistance value and also overrides any net resistance set by the *read_parasitics* command.

You can also use the *set_resistance* command for nets at lower levels of the design hierarchy. You can specify these nets as "BLOCK1/BLOCK2/NET_NAME".

To view resistance values, use the *report_net* command.

To remove a resistance value, use the *remove_resistance* command. To reset all back-annotated resistance values in a design, use the *reset_design* command.

Examples

The following example sets a resistance of 200 units to nets "a" and "b."

```
pt_shell> set_resistance 200 {a,b}
```

In the following example, a resistance of 300 units is set on net "U1/U2/NET3."

```
pt_shell> set_resistance 300 U1/U2/NET3
```

In the following example, the back-annotated resistance on net "U1/U2/NET3" is removed.

```
pt_shell> remove_resistance {get_nets U1/U2/NET3}
```

See Also

- find
- link
- remove_resistance
- report_net
- report_timing
- reset_design
- set_drive

---

set_retention

Defines the UPF retention strategy for the power domains in the design.

Syntax

```
status set_retention
   -domain domain_name
   [-no_retention]
   [-retention_power_net supply_net_name]
   [-retention_ground_net supply_net_name]
```

Feedback

```
[-retention_supply_set supply_set_name]
[-save_signal save_signal_and_save_sense]
[-restore_signal restore_signal_and_restore_sense]
[-elements object_list]
retention_name
```

Data Types

```
domain_name                 string
supply_net_name             string
supply_set_name             string
save_signal_and_save_sense    string
restore_signal_and_restore_sense     string
object_list                 list
retention_name              string
```

Arguments

`-domain` *domain_name*

Specifies the power domain name that this UPF retention strategy is applied to.

`-no_retention`

Specifies that the specified objects in the power domain does not have retention.

`-retention_power_net` *supply_net_name*

Specifies the retention power net for the retention cells that are under this UPF retention strategy.

`-retention_ground_net` *supply_net_name*

Specifies the retention ground net for the retention cells that are under this UPF retention strategy.

`-retention_supply_set` *supply_set_name*

Specifies the supply set whose power and ground function associations need to be used as the retention power and retention ground nets respectively. This argument is mutually exclusive with *-retention_power_net* and *-retention_ground_net*.

`-save_signal` *save_signal_and_save_sense*

Specifies the save signal net and the save signal sense for the retention cells under this retention strategy. Use the following syntax: {*save_signal* high|low}.

`-restore_signal` *restore_signal_and_restore_sense*

Specifies the restore signal net and the restore sense for the retention cells under this retention strategy. Use the following syntax: {*restore_signal* high|low}.

`-elements object_list`

Specifies the objects to which this UPF retention strategy is applied. The objects can be hierarchical cells, leaf cells, hdl blocks, nets.

retention_name

Specifies the UPF retention strategy name. The retention strategy name should be unique within the specified power domain.

Description

This command defines the UPF retention strategy on the specified power domain under which to map the unmapped sequential cells to retention cells.

PrimeTime uses the *set_retention* command to build virtual power ground connectivity. The command creates explicit connection for retention(backup) power and ground nets to power and ground pins of retention cells. The *-no_retention* option on specific elements takes precedence over domain-wide strategy.

If the *-elements* option is not specified, the retention strategy is applied to all of the unmapped sequential cells under the power domain. If the *-elements* option is specified, the UPF retention strategy is applied to all the unmapped sequential cells that are under the objects from *-elements*.

If neither the *-retention_supply_set* argument nor *-retention_power_net/-retention_ground_net* is specified, the retention power and ground nets are automatically set to the power and ground functions of the *default_retention* supply set handle of the power domain, for which the strategy is being defined.

For a list of UPF commands, use this command:

```
prompt>  help upf
```

Examples

The following example shows how to define a UPF retention strategy in the specified power domain PD1. Power domain PD1 is defined on instance shutdown_inst.

```
prompt> set_retention retention_1 -domain PD1 \\
          -retention_power_net PN1 \\
          -retention_ground_net GN1 \\
```

In the following example, it shows how to define a UPF retention strategy in the objects under the specified power domain.

```
prompt> set_retention retention_2 -domain PD1 \\
        -retention_power_net PN1 \\
        -retention_ground_net GN1 \\
        -elements shutdowm_inst/mid_inst \\
```

The following example shows how to define a UPF retention strategy using a supply set.

```
prompt> set_retention retention_2 -domain PD1 \\
        -retention_supply_set ret_supply_set \\
        -elements shutdown_inst/mid_inst/R_reg
```

The following example shows how a UPF retention strategy can be defined without using a supply set or supply nets.

```
prompt> set_retention retention_2 -domain PD1 \\
        -elements shutdown_inst/mid_inst/R_reg
```

See Also

- create_power_domain
- create_supply_net
- create_supply_set
- report_power_pin_info
- set_retention_control

---

set_retention_control

Defines the UPF retention control signals for the defined UPF retention strategy.

Syntax

status *set_retention_control*

```
-domain power_domain
-save_signal save_signal_and_save_sense
-restore_signal restore_signal_and_restore_sense
retention_strategy_name
```

Data Types

power_domain	string
save_signal_and_save_sense	string
restore_signal_and_restore_sense	string
retention_strategy_name	string

Arguments

`-domain` *power_domain*

Specifies the power domain name that this UPF retention strategy that is applied to.

`-save_signal` *`save_signal_and_save_sense`*

Specifies the save signal net and the save signal sense for the retention cells under this retention strategy. Use the following syntax: {*save_signal* high|low}.

`-restore_signal` *`restore_signal_and_restore_sense`*

Specifies the restore signal net and the restore sense for the retention cells under this retention strategy. Use the following syntax: {*restore_signal* high|low}.

`retention_strategy_name`

Specifies the UPF retention strategy name. The retention strategy should have already been defined using *set_retention* command

Description

This command defines the UPF retention control signals and control signal sense for this retention strategy of this power domain. The specified control signals are applied to the retention cells under the associated retention strategy.

PrimeTime uses additional information about the retention strategy provided by this command to properly match retention strategies to individual leaf cells. Retention strategy is used for deriving PG connectivity from UPF description. When multiple retention strategies match a retention cell, the last one entered is used.

Examples

The following example shows how to define the retention control signals on the defined UPF retention strategy *retention_1* of the power domain *PD1*.

```
prompt> set_retention_control retention_1 \\
         -domain PD1 \\
         -save_signal {save_net high} \\
         -restore_signal {restore_net low}
```

The following example shows how to find which retention strategy was used on which retention register to derive backup PG connectivity:

```
foreach_in_collection c [sort_collection
  [get_cells -hier * -filter "upf_retention_strategy != {}"] full_name] {
    echo [format
      {  Cell  %-8s  retention %s} [get_attribute $c full_name]
        [get_attribute $c upf_retention_strategy]]
}
```

See Also

- set_retention

Feedback

set_retention_elements

Defines a set of elements for a UPF retention strategy in the design.

Syntax

```
status set_retention_elements
   [-elements object_list]
   retention_element_name
```

Data Types

```
object_list              list
retention_element_name   string
```

Arguments

`-elements object_list`

Specifies the objects to which this UPF retention strategy is applied. The objects can be hierarchical cells, leaf cells, hdl blocks, nets.

`retention_element_name`

Specifies the UPF retention element name. This name is used for defining set of elements to be used in a retention strategy.

Description

This command works with *set_retention* command. It is used for defining a set of elements to be used in a retention strategy. Implementation tools use this command for error checking when applying a retention strategy. The PrimeTime tool does not perform any of this error checking; it only uses this element name when it directly refers to the element set in the *set_retetion* command.

For a list of UPF commands, use this command:

```
prompt>  help upf
```

Examples

The following example shows how to define a set of UPF retention elements.

```
prompt> set_retention_elements RET_ELEM -domain PD1 \\
        -elements {h2/r3}

prompt> set_retention my_retention -domain PD1 \\
        -retention_power_net PN1 \\
        -retention_ground_net GN1 \\
        -elements {RET_ELEM}
```

Feedback

See Also

- create_power_domain
- create_supply_net
- create_supply_set
- report_power_pin_info
- set_retention
- set_retention_control

---

set_routing_rule

Assigns a non-default or default routing rule to specific nets.

Syntax

integer *set_routing_rule*

```
[-rule rule | -default_rule | -no_rule]
[-min_routing_layer layer]
[-max_routing_layer layer]
[-min_layer_mode mode]
[-max_layer_mode mode]
[-min_layer_mode_soft_cost cost]
[-max_layer_mode_soft_cost cost]
[-min_layer_is_user true|false]
[-max_layer_is_user true|false]
[-clear]
[net_list]
```

Data Types

rule	string
layer	list
mode	string
cost	string
net_list	list

Arguments

`-rule rule`

Specifies the name of the non-default routing rule to be assigned to the nets.

The -rule, -default_rule, and -no_rule options are mutually exclusive; you can specify only one.

Feedback

`-default_rule`

Specifies that the default routing rule should be assigned to the nets.

The -rule, -default_rule, and -no_rule options are mutually exclusive; you can specify only one.

`-no_rule`

Removes the current routing rule, if any, from the nets. This allows the tool to automatically assign a non-default rule or the default rule to the nets.

The -rule, -default_rule, and -no_rule options are mutually exclusive; you can specify only one.

`-min_routing_layer` *`layer`*

Specifies the minimum routing layer for the nets. Only one layer can be specified.

`-max_routing_layer` *`layer`*

Specifies the maximum routing layer for the nets. Only one layer can be specified.

`-min_layer_mode` *`mode`*

Specifies the minimum routing layer mode for the nets. Valid values are unknown, extract_only, soft, allow_pin_connection, and hard. The default value is unknown.

`-max_layer_mode` *`mode`*

Specifies the maximum routing layer mode for the nets. Valid values are unknown, extract_only, soft, allow_pin_connection, and hard. The default value is unknown.

`-min_layer_mode_soft_cost` *`cost`*

Specifies the minimum routing layer mode soft cost for the nets. Valid values are unknown, low, medium, and high. The default value is unknown.

`-max_layer_mode_soft_cost` *`cost`*

Specifies the maximum routing layer mode soft cost for the nets. Valid values are unknown, low, medium, and high. The default value is unknown.

`-min_layer_is_user true|false`

Indicates whether or not the minimum routing layer is user-generated. True indicates it is user-generated. False indicates it is tool-generated. Default value is true.

Feedback

`-max_layer_is_user true|false`

Indicates whether or not the maximum routing layer is user-generated. True indicates it is user-generated. False indicates it is tool-generated. Default value is true.

`-clear`

Clears the nets' values. This resets the nets' routing rule, min/max routing layers, min/max layer modes, and min/max soft costs back to their default values. If this option is specified along with other options, the nets' values are cleared first and then set to the new values specified by the other options.

net_list

Lists of nets to which a routing rule is assigned.

Description

This command assigns a non-default rule, default routing rule, or no rule to specific nets. This command also assigns minimum and maximum routing layer, mode, and cost constraints to the nets.

Examples

The following example, assigns a non-default rule named *WideMetal* to a net named *B_CLK*.

```
prompt> set_routing_rule {B_CLK} -rule WideMetal
```

See Also

- reset_routing_rule
- report_routing_rules

---

set_rtl_activity_file

Specify RTL activity file for power analysis

Syntax

int *set_rtl_activity_file*

```
-file_type format_name
-strip_path strip_path
-path path
[-systemverilog]
[-time window_list]
file_name
```

Feedback

Data Types

format_name	string
strip_path	string
path	*string*
window_list	*list*
file_name	*string*

Arguments

`-file_type` *format_name*

Specifies file type of the given activity file. Supported format is fsdb/vcd/saif

`-strip_path` *strip_path*

Prefix to strip out from signal names of activity file.

`-path` *path*

Specifies a relative path from the current design to the hierarchical low-level design for which the SAIF file has been created. By default, absolute path names are used.

-systemverilog

Specifies the language format of names in the fsdb as systemverilog. Without this option, default language format is verilog.

`-time` *window_list*

Specifies time windows in which the activities are counted for power calculation. The option accepts an even number list of float values in increasing order. For example, *-time {10 20 50 70}* specifies the time window [10ns, 20ns] and [50ns, 70ns]. The default time window is the whole simulation time in FSDB/VCD.

file_name

Specifies the switching activity file name to be read.

Description

The *set_rtl_activity_file* command specify the switching activity file generated by RTL simulation. This command attempts to find the name mapping between the objects in activity file and in the gate level netlist automatically.

Examples

The following example shows how to specify systemverilog RTL FSDB for power analysis.

```
pwr_shell> set_rtl_activity_file -file_type fsdb -strip_path tb/top
 -systemverilog -time {10 20} rtl.fsdb
```

The following example shows how to specify RTL SAIF file

```
for power analysis.

pwr_shell> set_rtl_activity_file -file_type saif -strip_path top rtl.saif
```

See Also

- compute_metrics
- set_power_scenario
- set_prepower_options

---

set_rtl_architect_shell

Specify LEF technology file for NDM design database

Syntax

int *set_rtl_architect_shell*

```
        file_name
```

Data Types

```
file_name       string
```

Arguments

file_name

Specifies the location of the RTL Architect shell.

Description

The *set_rtl_architect_shell* command specifies location of the RTL Architect shell.

Examples

The following example shows how to specify RTL Architect shell.

```
pwr_shell> set_rtl_architect_shell full_path_to/rtl_shell
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs

- set_synth_phy_tech_file
- set_synth_output_dir
- set_presynth_options
- set_synth_options
- set_postsynth_options

---

set_rtl_export_data_options

Specify RTL export data options

Syntax

int *set_rtl_export_data_options*

```
[-include_libraries lib_type]
[-remove_existing_workspace]
```

Data Types

```
lib_type        string
```

Arguments

`-include_libraries lib_type`

Indicates what libraries to copy to the workspace. Currently only ‘design’ and ‘none’ is supported. Setting this to ‘design’ will copy the design library to the workspace. The default is ‘none’, which will not copy any libraries to the workspace.

-remove_existing_workspace

Removes any existing workspace and creates a new one. The default is not to remove any existing workspace. It reuses the workspace directory, thus overwrites on any existing content.

Description

The *set_rtl_export_data_options* command sets the options for the export_power_data or compute_metrics command that creates the workspace.

Examples

The following example copies the design library to the workspace.

```
pwr_shell> set_rtl_export_data_options -include_libraries {design}
```

```
The following example removes any existing workspace directory and
 creates a new one.

pwr_shell> set_rtl_export_data_options -remove_existing_workspace
```

See Also

- compute_metrics

---

set_rtl_to_gate_name

Sets the name mapping between the RTL and gate-level objects. Use this mapping to read the RTL backward SAIF file or VCD file for power estimation.

Syntax

integer *set_rtl_to_gate_name*

```
-rtl rtl_name
-gate gate_name
-substitute substitute
-inverted
```

Data Types

rtl_name	string
gate_name	string
substitute	string

Arguments

`-rtl` *rtl_name*

This option provides the name of a RTL object, whose name has changed after synthesis. The RTL object appears in the RTL backward SAIF or VCD file.

`-gate` *gate_name*

This option provides the name of gate-level object, for the corresponding RTL object. The name of the RTL object in the RTL backward SAIF or VCD file might change after synthesis. The *gate_name* is the new gate-level name of the RTL object after synthesis. This RTL object can be mapped to multiple gate-level objects due to replication.

`-substitute` *substitute*

This option directs the tool to substitute all the *from_string* occurrences in the RTL VCD or SAIF file with the *to_string*. Specify as *-substitute {from_string to_string}*.

Feedback

`-inverted`

This option specifies that the signal in the RTL activity file gets inverted when mapped to the gate-level design. You can use this option only when the object specified with the *-gate* option is a net or a pin.

Description

After synthesis, the names of the RTL objects might change due to which the *read_saif* or *read_vcd* command is not able to map the names present in the RTL SAIF or VCD file to the gate-level objects. Due to this, many of the nets, ports, and pins might not have the user-defined switching activity values, resulting in inaccurate power numbers.

To avoid this situation, the *set_rtl_to_gate_command* command allows you to set the mapping between the RTL and gate-level names after you have completed the RTL simulation only. You can set the RTL and gate-level names by using the *-rtl* and *-gate* options, respectively.

During the *read_saif* and *read_vcd* command, if the tool is unable to find the gate-level object matching the RTL object name, it looks for the corresponding gate-level name; however, if the gate-level name is provided using the *set_rtl_to_gate_command* command, the tool tries to find the corresponding gate-level object.

Note: Specify the full name of the object with respect to the current instance to the *gate_name* argument. Similarly, specify the the full name as it appears in the backward RTL SAIF or VCD file to the *rtl_name* argument.

When you use the *-substitute* option, any signal name in the RTL VCD or SAIF is scanned only one time. The substitutions are not applied recursively.

Examples

The following example provides the mapping between RTL and gate-level objects.

```
pt_shell> set_rtl_to_gate_name -rtl reg_i[1] -gate reg_i_1
```

The following example provides the mapping for an RTL level signal to a gate-level pin, and instructs the tool to invert the signal found in the RTL activity file.

```
pt_shell> set_rtl_to_gate_name -rtl test_signal \\
             -gate test_signal_reg/Q  -inverted
```

See Also

- unset_rtl_to_gate_name
- reset_rtl_to_gate_name
- report_switching_activity
- read_saif

set_rule_severity

Sets the severity of a rule.

Syntax

Boolean *set_rule_severity*

```
severity_value
rule_list
```

Data Types

```
severity_value    string
patterns          list
```

Arguments

severity_value

Specifies the new severity value. Valid values are *info*, *warning*, *error*, or *fatal*.

rule_list

Lists the rule names or patterns. Patterns can include the wildcard characters "*" and "?". Patterns can also include collections of type rule.

Description

Modifies the severity of specified rules. You can change the severity of built-in or user-defined rules. Severity is shown in the output of the *check_constraints* command. You can get the severity of a rule using *get_attribute $rule severity*.

Examples

The following example sets the severity for rule *EXD_0009* to "error".

```
pt_shell> set_rule_severity error EXD_0009
```

Feedback

See Also

- check_constraints
- get_attribute

---

set_scenario_case_analysis

Specifies a constant logic value for a list of ports or pins for specific SMS scenarios.

Syntax

```
status set_scenario_case_analysis
  value
  port_or_pin_list
  [-sms_scenarios sms_scenarios_list]
```

Data Types

value	string
port_or_pin_list	list
sms_scenarios_list	collection

Arguments

value

Specifies a constant logic value or a transition type assigned to the given ports or pins. These are the valid values:

- *0* or *zero* - Constant logic 0
- *1* or *one* - Constant logic 1

port_or_pin_list

Specifies the ports or pins affected by the case analysis setting.

Description

Scenario case analysis is a limited form of case analysis settings for SMS scenarios. Scenario case analysis settings can affect clock latencies and datapaths by preventing arrivals from some scenarios from propagating through certain arcs.

Case analysis specifies a list of ports or pins that have a constant logic value (*0*, *1*).

When case analysis is specified, this value is propagated through the network where the constant value controls the traversed logic. For example, if you specify a constant value of 0 at an input of a NAND gate, the output of the NAND gate has a constant value of 1. This value is then propagated further to all cells driven by this output.

Feedback

Scenario case analysis can be used to affect signal propagation especially when cell arcs have "when" conditions. Arcs may become blocked for arrival signal propagation in the given SMS scenario.

Arcs disabled by scenario can be reported with *report_disable_timing*.

The current *set_scenario_case_analysis* settings can be reported using the *report_case_analysis* command.

Scenario case settings can be removed using the \Bremove_scenario_case_analysis command.

Scenario case analysis has limitations compared to the regular *set_case_analysis* command:

1. Scenario case analysis does not affect design topology, including combinational loop breaking or determination of which registers are clocked by which clock. 2. Scenario case analysis does not affect slew propagation or delay calculation on the design. Slews are propagated through arcs in a given scenario even if the arc is blocked for arrival propagation in the same scenario. 3. Scenario case analysis does not affect mode settings of cells in the design. 4. *report_case_analysis* can show the user specified settings, but does not show propagated scenario case analysis.

Examples

The following command sets the IN1 port to constant logic 0 for a specific scenario:

```
pt_shell> set_case_analysis 0 IN1 -sms_scenarios [get_sms_scenarios
 -corner {A}]
```

If there are cells downstream of IN1, some arcs on those cells may become blocked for clock latency and data arrival propagation in the specified scenario. If the cells are part of the clock network, clock paths reported by *report_timing -path_type full_clock_expanded* may take different paths because of the setting.

See Also

- remove_scenario_case_analysis
- report_case_analysis
- report_disable_timing

---

set_scope

Specify the current UPF scope. Return the current UPF scope prior to the execution of this command as a full path string relative to the current design top if successful and null string if it fails.

Feedback

Syntax

string *set_scope*

`[instance]`

Data Types

`instance` `string`

Arguments

`instance`

Specifies the working instance in pt_shell. If the *instance* option is not specified, the focus returns to the top level of the current design. If *instance* is ".", pt_shell returns to the working instance. If *instance* is "..", the context is moved up one level in the instance hierarchy. If *instance* begins with "/", pt_shell returns to the working instance of the design whose name is after the "/". More complex examples of *instance* arguments are described below in EXAMPLES.

Description

This command sets the UPF scope in pt_shell. Functionally, this command is the subset of the *current_instance* command, but with different return value.

- If no *instance* argument is specified, the UPF scope of pt_shell is returned to the top level of the hierarchy.
- If *instance* is ".", the UPF scope is returned and no change is made.
- If *instance* is "..", the UPF scope is moved up one level in the design hierarchy.
- If *instance* is a valid cell at the current level of hierarchy, the UPF scope is moved down to that level of the design hierarchy.
- Multiple levels of hierarchy can be traversed in a single call to *set_scope* by separating multiple cell names with slashes.
- If *instance* begins with "/", the tool returns to the working instance under current design top instance whose name is after the "/". Basically, the "/" in the beginning of instance name is interpreted as design top instance. If *instance* is "/", the tool returns the current design top instance. load_upf -scope implicitly changes the design top instance (and design top model)to the -scope value.

  For example, the *set_scope U1/U2* command sets the UPF scope down two levels of hierarchy if both cells exist at the current levels in the design hierarchy.
- The ".." directive can also be nested in complex *instance* arguments.

  For example, the *set_scope "../../MY_INST"* command attempts to move the context up two levels of hierarchy, then down one level to the "MY_INST" cell.

NOTE: The *set_scope* command does not work on leaf cells. If you attempt to run this command on a leaf cell, an error occurs.

See Also

- current_instance
- current_design

---

set_sense

Specifies unateness propagating forward for pins with respect to clock source.

Syntax

status *set_sense*

```
[-type clock | data]
[-non_unate]
[-rise]
[-fall]
[-clock_fall]
[-clock_rise]
[-primary]
[-generated]
[-positive]
[-negative]
[-stop_propagation]
[-clock_leaf]
[-pulse pulse_type]
[-clocks clock_list]
object_list
```

Data Types

```
pulse_type      string
clock_list      list
object_list      list
```

Arguments

`-type clock | data`

Specifies whether the sense is applied to clock or data networks. If you do not specify this option, the default is *clock*. If you specify *data*, you must also use the *-stop_propagation* and *-clocks* options.

`-non_unate`

Launches both the positive-unate sense and the negative-unate sense of the clock from the clock source. If you use the *-non_unate* option, you must also use

the *-clocks* option. Please note that this option is only valid for clock source pins/ports.

`-rise`

Specifies rising clock sense. Please note that this option is only valid for clock source pins/ports.

`-fall`

Specifies falling clock sense. Please note that this option is only valid for clock source pins/ports.

`-clock_fall`

Specifies that the signal is coming from the falling edge of the clock. If not specified, the command uses the falling edge for *-fall*. Please note that this option is only valid for clock definition points.

`-clock_rise`

Specifies that the signal is coming from the rising edge of the clock. If not specified, the command uses the rising edge for *-rise*. Please note that this option is only valid for clock definition points.

`-primary`

Specifies primary clock signal type, primary is the default. Please note that this option is only valid for clock definition points.

`-generated`

Specifies generated clock signal type. Please note that this option is only valid for clock definition points.

`-positive`

Applies positive unateness to all pins in the *object_list* with respect to clock source. If you use the *-positive* option, you cannot use the *-negative* or *-pulse* options.

`-negative`

Applies negative unateness to all pins in the *object_list* with respect to clock source. If you use the *-negative* option, you cannot use the *-positive* or *-pulse* options.

`-stop_propagation`

Stops the propagation of specified clocks in the *clock_list* from the specified pins or cell timing arcs in the *object_list*. Only a clock used as a clock is stopped if you specify the *-type clock* option. To stop propagation for both clock-as-clock and clock-as-data, you need to use a *set_sense -type clock* command with a

set_sense -type data command. You cannot use the *-stop_propagation* option with the *-positive*, *-negative*, or *-pulse* options.

`-clock_leaf`

Stops the propagation of specified clocks in the *clock_list* from the specified pins in the *object_list*. Also, it specifies the pin as a clock consumer, so that the clock branches reaching this pin will be considered as clock consumer as well. In other words, the attribute *"is_clock_used_as_clock"* will be true for all the pins of the clock branches that reach the clock leaf pin. You cannot use the *-clock_leaf* option with the *-positive*, *-negative*, *-stop_propagation* or *-pulse* options.

`-pulse` *`pulse_type`*

Specifies the type of pulse clock applied to all pins in the *object_list* with respect to clock source. The possible values for the *pulse_type* argument are

- *rise_triggered_high_pulse*
- *fall_triggered_high_pulse*
- *rise_triggered_low_pulse*
- *fall_triggered_low_pulse*

If you use the *-pulse* option, you cannot use the *-positive* or *-negative* options.

`-clocks` *`clock_list`*

Specifies a list of clock objects to be applied with the given unateness that is placed on all pin objects in the *object_list*. If you do not specify the *-clocks* option, all clocks passing through the given pin objects are considered.

`object_list`

List of pins, ports, or cell timing arcs with specified unateness to propagate. The timing arcs object can be used with the *-stop_propagation* option only.

Description

This command modifies the waveform of the clock in two distinct ways: 1. You can use this command to restrict unateness at pin (to positive or negative unate) with respect to clock source. However, the specified unateness only applies within the non-unate clock network. In this case, user-defined sense propagates forward from the given pins.

1. You can specify the waveform of the clock at its definition point. This option is only valid for the clock sources of the clock. You can specify the waveform to be launched from the clock source as positive, negative or non-unate. You can also specify whether the clock is only used for source latency computation of the generated clocks. This can be specified by the *-generated* option.

Feedback

If you use the *-clocks* option, only the specified clock domains are applied. Otherwise, all clocks passing through the given pin objects are considered.

PrimeTime issues warning messages if the specified sense on given pins cannot be respected in case there is predefined unateness for given pins. A hierarchical pin is not supported.

In addition, you can completely stop propagation of certain clocks in clock networks or data networks by using the *-stop_propagation* option along with the *-type* option.

To undo the *set_sense* command, use the *remove_sense* command.

Examples

The following example specifies a positive unateness for the XOR/Z pin with respect to the CLK1 clock.

```
pt_shell> set_sense -positive -clocks [get_clocks CLK1] XOR/Z
```

The following example specifies negative unateness for the MUX/Z pin for all clocks.

```
pt_shell> set_sense -negative MUX/Z
```

The following example specifies a positive unate clock sense on primary clock CLK1 using a combination of -fall, -rise, -clock_fall, -clock_rise.

```
pt_shell> set_sense -primary -rise -clock_rise -clocks CLK1 inv2/Y
pt_shell> set_sense -primary -fall -clock_fall -clocks CLK1 inv2/Y
```

The following example specifies a positive unate waveform for the clock and a negative unate waveform for the clock when it is used for source latency computation.

```
pt_shell> set_sense -primary -fall -clock_fall -clocks CLK1 inv2/Y
pt_shell> set_sense -primary -rise -clock_rise -clocks CLK1 inv2/Y
pt_shell> set_sense -generated -fall -clock_rise -clocks CLK1 inv2/Y
pt_shell> set_sense -generated -rise -clock_fall -clocks CLK1 inv2/Y
```

See Also

- remove_sense

---

set_si_aggressor_exclusion

Sets the given nets to be exclusive while switching in the given direction, when they are aggressors to the same victim net.

Syntax

integer *set_si_aggressor_exclusion*

```
[-number_of_active_aggressors n]
[-rise]
[-fall]
anets
```

Data Types

```
anets        list
n            integer
```

Arguments

`-number_of_active_aggressors n`

Specifies the maximum number of aggressors among the given list of exclusive aggressors that can be active at a given instant. This option is used to set n-hot or n-cold behavior, where only the *n* nets are active(hot) or quiet(cold) at a given instant. If this option is not specified, the default value of *n=1* is used; i.e only one of the given exclusive aggressors is considered active at a given time.

`-rise`

Specifies the aggressor nets to be exclusive while switching in the rise direction. If neither the *-rise* nor the *-fall* options are specified, the aggressor *anets* nets are considered exclusive in the directions of both the *-rise* and *-fall* options.

`-fall`

Specifies the aggressor nets to be exclusive while switching in the fall direction. If neither the *-rise* nor *-fall* options are specified, the aggressor *anets* net are considered exclusive in in the directions of both the *-rise* and *-fall*.

`anets`

Specifies the list of nets which are exclusive when they are aggressors to the same victim net.

Description

The *set_si_aggressor_exclusion* command sets the aggressor nets to exclusive while switching in the given direction at the same time. Among all combinations of active aggressors allowed by the *-number_of_active_aggressors option, only the combination which produces the worst alignment are considered for crosstalk analysis.*

Only the aggressor nets specified in the *anets* list are considered exclusive. If a victim net is specified in the *anets* list, it is not considered exclusive with any of its aggressor nets.

To view which aggressor nets were considered active and which aggressor nets were set to quiet, use the *report_delay_calculation* or *report_noise_calculation* commands on the victim net. The quiet aggressor nets are reported to be screened due to aggressor exclusion.

Feedback

These exclusive commands are independent of parasitics, so they are applied even before reading in parasitics.

Note that application of exclusive aggressors cannot be done incrementally; next update_timing and update_noise would be a full update.

The command *report_si_aggressor_exclusion* is used to check which groups are set by the *set_si_aggressor_exclusion* command.

Examples

The following example shows how the nets *SCAN_LOGIC** are set to be exclusive while switching in the rise direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets SCAN_LOGIC*] -rise
1
```

The following example shows how the nets *LCO_LOGIC** are set to be exclusive while switching in the fall direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets LCO_LOGIC*] -fall
1
```

The following example shows how only 3 of the nets *LOGIC_BUS** considered to be active while switching in the rise direction.

```
pt_shell> set_si_aggressor_exclusion [get_nets LOGIC_BUS*] -rise
-number_of_active_aggressors 3
1
```

See Also

- remove_si_aggressor_exclusion
- report_si_aggressor_exclusion
- report_delay_calculation
- report_noise_calculation
- si_analysis_logical_correlation_mode
- set_si_delay_analysis
- set_si_noise_analysis

---

set_si_delay_analysis

Sets coupling information on nets for crosstalk analysis.

Feedback

Syntax

integer *set_si_delay_analysis*

```
[-ignore_arrival inets]
[-exclude]
[-victims vnets]
[-aggressors anets]
[-rise]
[-fall]
[-min]
[-max]
```

Data Types

inets	list
vnets	list
anets	list

Arguments

`-ignore_arrival inets`

Specifies a list of nets to be analyzed as infinite window. You cannot use this option with the *-exclude*, *-victims*, or *-aggressors* options. The physical exclusivity analysis for SI is considered for these nets when si_xtalk_use_physical_exclusivity_on_ignore_arrival_nets variable is set to true.

`-exclude`

Indicates that nets specified as a *vnets* or *anets* variable are to be excluded from the crosstalk analysis as victim nets or aggressor nets, respectively. You cannot use this option with the *-ignore_arrival* option. When both the *-victims vnets* and *-aggressors anets* options are applied, all cross capacitances between the *vnets* and *anets* variables are excluded, when *vnets* are victims and *anets* are aggressors.

`-victims vnets`

Specifies the list of nets on which the *-exclude* option information is applied as a victim. You cannot use this option with the *-ignore_arrival* option. If you use the *-victims* option, you must use the *-exclude* option. When used with the *-aggressors* option, the *-victims* option excludes the cross capacitances between the victim nets (*vnets*) and the aggressor nets (*anets*).

`-aggressors anets`

The list of nets on which the *-exclude* option information is applied as an aggressor. You cannot use this option with the *-ignore_arrival* option. If you use the *-aggressors* option, you must use the *-exclude* option. When used with the *-victims* option, the *-aggressors* option excludes the cross capacitances between the victim nets (*vnets*) and the aggressor nets (*anets*).

`-rise`

Excludes a list of nets for victim rising. If you use the *-rise* option, you must use the *-exclude* option.

`-fall`

Excludes a list of nets for victim falling. If you use the *-fall* option, you must use the *-exclude* option.

`-min`

Excludes a list of nets for min path analysis. If you use the *-min* option, you must use the *-exclude* option.

`-max`

Excludes a list of nets for max path analysis. If you use the *-max* option, you must use the *-exclude* option.

Description

Sets coupling information on nets for crosstalk analysis.

The *set_si_delay_analysis* command allows you to set some net as infinite window as both aggressor and victim. This command has no default. To remove the result of this command, use the *remove_si_delay_analysis* command.

The *set_si_delay_analysis* command and the *-exclude* option allows you to select nets to be excluded from crosstalk analysis. By default, all nets with coupling capacitances (non-filtered) are included in crosstalk analysis as both victim and aggressor. This command allows the nets to be excluded as victim (by using the *-victim* option) or aggressor (by using the *-aggressors* option). When you use both the *-victims* and *-aggressors* options, the command excludes the relationship between the specified victim nets (*vnets*) and aggressor nets (*anets*); this is referred to as "pair-wise exclusion."

By using the *set_si_delay_analysis* command with the *-ignore_arrival* option, you can set the *inets* as infinite window. When *inets* is analyzed as victim, the timing relationship between all the nets in this coupling cluster is ignored, and all aggressors become active (even those driven by clock domains which are physically exclusive). When *inets* is analyzed as aggressor the arrival time of *inets* is ignored. The infinite window analysis could make the crosstalk calculation more pessimistic.

The *set_si_delay_analysis* command returns a *1* if successful and a *0* if unsuccessful.

To view the results of this command, use the *report_si_delay_analysis* command.

Examples

The following example shows that all nets described by *CLK_NET_** are excluded from crosstalk analysis as victim nets.

Feedback

```
pt_shell> set_si_delay_analysis -exclude -victims [get_nets CLK_NET*]
1
```

The following example shows how all nets in a design to be analyzed as infinite window.

```
pt_shell> set_si_delay_analysis -ignore_arrival [get_nets -hier *]
1
```

The following example shows how to ensure that the scan clock described by *SCN_CLK_** is not effecting the main clock network described by *CLK_NET_**, and vice versa.

```
pt_shell> set_si_delay_analysis -exclude -victims \\
[get_nets SCN_CLK_*] \\
-aggressors [get_nets CLK_NET*]
pt_shell> set_si_delay_analysis -exclude -aggressors \\
[get_nets SCN_CLK_*] -victims [get_nets CLK_NET*]
1
```

See Also

- read_parasitics
- remove_si_delay_analysis
- report_si_delay_analysis
- si_enable_analysis

---

set_si_delay_disable_statistical

Disables composite aggressor statistical analysis on nets for crosstalk analysis.

Syntax

int *set_si_delay_disable_statistical*

```
        dnets
```

Data Types

dnets list

Arguments

dnets

A list of nets in the current design for which the composite aggressor statistical analysis is disabled.

Feedback

Description

If selected in composite aggressor group for crosstalk analysis, *dnets* specified by this command are not treated statistically. If they are not selected into composite aggressor group, this command has no effect.

To remove the result of this command, use the *remove_si_delay_disable_statistical* command.

Examples

The following example shows how to disable net from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_delay_disable_statistical [get_nets LOGIC1]
1
```

See Also

- remove_si_delay_disable_statistical
- report_si_delay_analysis

---

set_si_noise_analysis

Sets coupling information on nets for noise analysis.

Syntax

int *set_si_noise_analysis*

```
[-ignore_arrival inets]
[-exclude]
[-victims vnets]
[-aggressors anets]
[-above]
[-below]
[-low]
[-high]
```

Data Types

inets list *vnets* list *anets* list

Arguments

`-ignore_arrival inets`

Specifies a list of nets to be set as infinite window. When *inets* are analyzed as victims, all of their aggressors are set with infinite window. When *inets* are analyzed as aggressors, they are set as aggressors with arrival time ignored.

You cannot use this option with the *-exclude*, *-victims*, *-aggressors*, *-high*, *-low*, *-above*, or *-below* option.

`-exclude`

Indicates that nets specified as *vnets* or *anets*) are to be excluded from the noise analysis as victim nets or aggressor nets, respectively. You cannot use this option with the *-ignore_arrival* option. When both the *-victims vnets* and *-aggressors anets* options are applied, the noise between *vnets* and *anets* is not analyzed, when *vnets* are victims and *anets* are aggressors.

`-victims` *`vnets`*

Specifies the list of nets on which *-exclude* information is applied as a victim. You cannot use this option with the *-ignore_arrival* option. If you use the *-victims* option, you must use the *-exclude* option. When used with the *-aggressors* option, the *-victims* option excludes the noise analysis between the victim nets (*vnets*) and the aggressor nets (*anets*).

`-aggressors` *`anets`*

Indicates the list of nets on which the *-exclude* option information is applied as an aggressor. You cannot use this option with the *-ignore_arrival* option. If you use the *-aggressors* option, you must use the *-exclude* option. When used with the *-victims* option, *-aggressors* excludes the noise analysis between the victim nets (*vnets*) and the aggressor nets (*anets*).

`-above`

Excludes a list of nets for victim above rail. If you use the *-above* option, you must use the *-exclude* option.

`-below`

Excludes a list of nets for victim below rail. If you use the *-below* option, you must use the *-exclude* option.

`-low`

Excludes a list of nets for low rail noise analysis. If you use the *-low* option, you must use the *-exclude* option.

`-high`

Excludes a list of nets for high rail noise analysis. If you use the *-high* option, you must use the *-exclude* option.

Description

Sets coupling information on nets for noise analysis.

The *set_si_noise_analysis* command allows you to exclude some net from noise analysis or set some net as infinite window as aggressor. This command has no default. To remove the result of this command, use the *remove_si_noise_analysis* command.

The *-exclude* option of the *set_si_noise_analysis* command allows you to select nets to be excluded from noise analysis. By default, all nets with coupling capacitances (non-filtered) are included in noise analysis as both victim and aggressor. This command allows the nets to be excluded as victim (by using the *-victim* option) and/or aggressor (by using the *-aggressors* option). When you use both the *-victims* and *-aggressors* options, the command excludes the noise analysis between the specified victim nets (*vnets*) and aggressor nets (*anets*); this is referred to as "pair-wise exclusion."

By using the *set_si_noise_analysis* command with the *-ignore_arrival* option, you can set the *inets* as infinite window as aggressor if they are analyzed as aggressors. When *inets* are analyzed as noise victims, all of their aggressors are infinite window.

The *set_si_noise_analysis* command returns a *1* if successful and a *0* if unsuccessful.

To view the results of this command, use the *report_si_noise_analysis* command.

Examples

The following example shows that all nets described by *CLK_NET_** are excluded from noise analysis as victim nets.

```
pt_shell> set_si_noise_analysis -exclude -victims [get_nets CLK_NET*]
1
```

The following example shows that when all nets described by *SOME_LOGIC** are analyzed as aggressors, they are set as infinite window as aggressors in noise analysis.

```
pt_shell> set_si_noise_analysis -ignore_arrival [get_nets SOME_LOGIC*]
1
```

The following example shows how to ensure that the scan clock described by *SCN_CLK_** is not effecting the main clock network described by *CLK_NET_**, and vice versa.

```
pt_shell> set_si_noise_analysis -exclude -victims \\
           [get_nets SCN_CLK_*] \\
           -aggressors [get_nets CLK_NET*]
pt_shell> set_si_noise_analysis -exclude -aggressors \\
           [get_nets SCN_CLK_*] -victims [get_nets CLK_NET*]
1
```

See Also

- read_parasitics
- remove_si_noise_analysis

Feedback

- report_si_noise_analysis
- si_enable_analysis

---

set_si_noise_disable_statistical

Disables composite aggressor statistical analysis on nets for noise analysis.

Syntax

int *set_si_noise_disable_statistical*

```
        dnets
```

Data Types

dnets	list

Arguments

dnets

A list of nets in the current design for which the composite aggressor statistical analysis is disabled.

Description

If selected in composite aggressor group for noise analysis, *dnets* specified by this command are not treated statistically. If they are not selected into composite aggressor group, this command has no effect.

To remove the result of this command, use the *remove_si_noise_disable_statistical* command.

Examples

The following example shows how to disable net from statistical analysis when considered as a composite aggressor.

```
pt_shell> set_si_noise_disable_statistical [get_nets LOGIC1]
1
```

See Also

- remove_si_noise_disable_statistical
- report_si_noise_analysis

Feedback

set_signal_em_analysis_options

Sets the options for signal electromigration analysis.

Syntax

status *set_signal_em_analysis_options*

```
[-em_temperature temperature]
[-em_delta_temperature temperature]
```

Data Types

```
temperature        float
```

Arguments

`-em_temperature temperature`

Specifies temperature for AVG current limit lookup. Default temperature is parasitics temperature.

`-em_delta_temperature temperature`

Specifies delta temperature used in RMS current limit lookup. Default value is 5.0.

Description

Command to specify various analysis options for signal electromigration analysis in PrimePower. The options from this command controls how signal electromigration analysis behaves for specific analysis type.

Examples

The following example sets temperatures for signal electromigration analysis.

```
pt_shell> > set_signal_em_analysis_options -em_temperature 95 \
                        -em_delta_temperature 10
```

set_size_only

Sets the *size_only* attribute on leaf cells to prevent those objects from being removed during ECO fixing.

Syntax

status *set_size_only*

```
object_list
[value]
```

Feedback

Data Types

```
object_list     list
value           Boolean
```

Arguments

object_list

Specifies a collection of leaf cells, preventing those objects from being removed during ECO fixing.

Lists are not supported.

value

Sets the value, either *true* or *false*, for the *size_only* attribute of the specified objects. If no value is provided, the default is *true*. Use the value *false* to cancel the effects a previous *set_size_only* command.

Description

This command sets the *size_only* attribute on leaf cells to prevent those objects from being removed during ECO fixing by the *fix_eco_drc*, *fix_eco_power*, and *fix_eco_timing* commands. The specified objects must be in one of the target libraries or in a library already loaded into memory.

You can query the *size_only* attribute by using the *get_attribute* command. To cancel the effects of a *set_size_only* command, run the command again and specify the *value* as *false*. A *false* setting on an object overrides a *true* setting on the object.

In distributed multi-scenario analysis (DMSA), you should apply the *set_size_only* command in all scenarios using *remote_execute*.

Examples

The following commands specify that the cell "u1" is *size_only* and should not be removed during ECO.

```
pt_shell> set_size_only [get_cells {u1}]
1
pt_shell> get_attribute [get_cells {u1}] size_only
true
```

See Also

- fix_eco_drc
- fix_eco_power
- fix_eco_timing

Feedback

- get_attribute
- set_dont_touch

---

set_spacing_label_rule

Sets an intercell spacing constraint between reference cells that have been assigned labels with set_lib_cell_spacing_label command.

Syntax

status *set_spacing_label_rule*

```
-labels list_of_label_names
min_max_spec
```

Data Types

list_of_label_names	list
min_max_spec	integer

Arguments

`-labels` *list_of_label_names*

Specifies a list of exactly two intercell constraint labels previously defined by the set_lib_cell_spacing_label command.

min_max_spec

Specifies the range of illegal x-spacing values in units of sites. *min_max_spec* follows the format *{minx maxx}* where minx <= maxx. The spacing between the labeled edges cannot be in the range from minx to maxx placement sites.

Description

This command assigns intercell spacing constraints between reference cells that have been assigned labels with the set_lib_cell_spacing_label command. Cells with the given labels are prohibited from being placed adjacent to one another with a spacing between minx and maxx. The range is specified in units of placement sites.

Given two adjacent standard cells with an empty space in between, each label from the right side of the left cell is checked against each label from the left side of the right cell to see if any spacing rules are violated. Such pairwise checking of labels is applied to all adjacent cells to ensure placement legality.

A label typically represents some layout features inside a standard cell. A label is associated with the left or right boundary of a standard cell in the "North" orientation. When a cell is flipped, its associated labels are also flipped.

With intercell spacing rules specified, standard cells can be placed next to each other only when they do not violate the spacing rules.

The labels and spacing rules can be used to express spacing requirements that originate from mask rules and the layout of standard cells. The labels and spacing rules are stored in the library and are applied to all designs using the library.

Examples

In the following example, cells with labels "X" and "Y" are prohibited to abut (a spacing of 0 is prohibited).

```
prompt> set_spacing_label_rule -labels {X Y} {0 0}
```

In the following example, cells with labels "X" are prohibited to be spaced within range of 3 to 5 placement sites.

```
prompt> set_spacing_label_rule -labels {X X} {3 5}
```

See Also

- set_lib_cell_spacing_label

---

set_steady_state_resistance

Specifies the steady-state resistance for a library pin or port.

Syntax

status *set_steady_state_resistance*

```
[-above]
[-below]
[-low]
[-high]
res_value
object_list
```

Data Types

res_value	float
object_list	list

Arguments

`-above`

Specifies steady state resistance for above ground or power rail noise analysis region.

`-below`

Specifies steady state resistance for below ground or power rail noise analysis region.

`-low`

Specifies steady state resistance for ground rail noise.

`-high`

Specifies steady state resistance for power rail noise.

`res_value`

Specifies the steady state resistance value in units of resistance.

`object_list`

Specifies a list of library pins or ports.

Description

A steady-state driver of a net affects the size of crosstalk bumps on the net due to its loading effects on the net. To accurately calculate the characteristics of crosstalk noise bumps, the tool needs the steady-state I-V characteristics of the driver output.

Noise bumps can occur in the following regions: below low, above low, below high, and above high. You can specify a steady-state resistance for each region.

This command can be used in the absence of library-specified I-V characteristics, or to override the library-specified characteristics by replacing them with a steady-state resistance value.

The *report_noise_calculation* command shows whether noise immunity information was taken from the library or annotated by this command.

Examples

This example specifies a steady state resistance of 2.0 for above-the-ground rail noise for the Z pin of the AN4 library cell in the lsi_10k library:

```
pt_shell> set_steady_state_resistance -above -low  2.0 lsi_10k/AN4/Z
```

See Also

- remove_steady_state_resistance
- report_noise_calculation

Feedback

set_supply_net_probability

Sets the static probability annotation on selected supply nets. This probability affects average and time-based power analysis.

Syntax

```
int set_supply_net_probability
[-remove]
supply_nets
[static_prob]
```

Data Types

```
supply_nets          list
static_pro          float
```

Arguments

`-remove`

Removes the annotation on given supply nets.

supply_nets

Specifies the supply net or supply nets to apply the probability to. This argument can be a supply net name, a list of supply net names, a supply net obtained using the get_supply_nets com- mand, or a collection of supply nets obtained using the get_sup- ply_nets command.

static_prob

Specifies the probability that the supply net is powered. Use a value between 0 and 1.

This argument is optional. If not supplied, no annotation is performed and a description of the static probability on the supply nets is returned.

Description

Averaged power analysis uses the supply net probability to determine what fraction of the time the cells connected to this supply net are powered. Time-based power analysis uses the supply net probability to determine the cells connected to this supply net are powered on/off. This command can be used for any supply net irrespective of whether they are connected to a on chip switch cell or not.

Examples

The following examples show how to use the command:

```
pt_shell> get_supply_nets *
{"VDDI", "VDDM", "VDDIS", "VDD", "VDDMS", "VDDGS", "VDDG", "VSS"}
```

Feedback

```
pt_shell> set_supply_net_probability VDDI 0.25
{"VDDI" 0.25 annotated}

pt_shell> set_supply_net_probability [list VDDI VDD] 0.4
{"VDDI" 0.4 annotated} {"VDD" 0.4 annotated}

pt_shell> set_supply_net_probability [get_supply_nets *] 0.4
{"VDDI" 0.4 annotated} {"VDD" 0.4 annotated} ...

pt_shell> set_supply_net_probability [get_supply_nets VDDI] 0.4
{"VDDI" 0.4 annotated}
```

See Also

- create_supply_net
- update_power

---

set_switching_activity

Sets switching activity annotation on the selected nets, pins, ports, and cells of the current design.

Syntax

integer *set_switching_activity*

```
[-state_condition state_condition]
[-path_sources path_sources]
[-toggle_rate toggle_rate]
[-clock_derate derate_value]
[-glitch_rate glitch_rate]
[-static_probability static_probability]
[-rise_ratio rise_ratio]
[-period period_value]
[-base_clock clock]
[-type object_type_list]
[-hierarchy]
[-verbose]
[-force]
[-clock_domains clock_list]
[object_list]
```

Data Types

state_condition	string
path_sources	list
toggle_rate	float
derate_value	float
glitch_rate	float

```
static_probability          float
rise_ratio                  float
period_value                float
clock                       string
object_type_list            list
clock_list                  list
object_list                 list
```

Arguments

`-state_condition` *state_condition*

Specifies the state condition when annotating state-dependent toggle rate and glitch rate on pins or state-dependent static probabilities on cells. State-dependent toggle rates and glitch rates can be annotated when the internal power of the library cell pin is characterized with state-dependent power tables. State-dependent static probabilities can be annotated when the cell leakage power is characterized with state-dependent power tables. The state condition specified with this argument must be logically equivalent to a state condition in the internal or leakage power characterization. The state condition should be enclosed between " ". To set switching activity for the default state condition, specify the argument of *-state_condition* option as *default*.

`-path_sources` *path_sources*

Specifies the path sources when annotating path-dependent toggle rate and glitch rate on pins. This is used when the library cell pin has path-dependent internal power. The path sources specified with this argument must be the same as the path sources in the internal power characterization. When listing more than one pin in path sources, the pin names must be separated by a space and enclosed between " ". For example, if the pins in path sources are A and B, the argument for the *-path_sources* option must be "A B". Note: If you use the *-path_sources* option to specify the *path_sources value, then you must use the -state_condition option to specify the value of the path_sources.*

`-static_probability` *static_probability*

Specifies the value of the static_probability switching activity. The *static_probability* value represents the percentage of time the signal is at the logic state 1. For example, a *static_probability* of 0.25 indicates that the signal is in the logic state 1 for 25% of the time.

`-toggle_rate` *toggle_rate*

Specifies the value of the toggle rate switching activity. The rate is a floating point number that represents the number of 0->1 and 1->0 glitch free transitions, that the signal makes during a period of time. You can specify the period with the *-period* option. Alternatively, you can annotate a related clock with the *-base_clock* option, and the specified toggle rate is relative to the related clock period. If you specify the *-clock_domains* option, the related clock is chosen

based on which clock domain the object belongs. If it belongs to multiple clock domains, the faster clock is the related clock. If no clock domain is found for the object, the fastest clock in the design is the related clock.

`-clock_derate` *`derate_value`*

The *-clock_derate* option is an alternative to the *-toggle_rate* option if the *-base_clock* option is specified. The annotated toggle rate for an object is a factor of the toggle rate of the related clock of the object. The related clock is chosen based on the clock domain to which the object belongs. If it belongs to multiple clock domains, the fastest one is selected. If the object does not belong to any clock domain, the related clock is set to the fastest clock in the design.

`-glitch_rate` *`glitch_rate`*

Specifies the value of the glitch rate switching activity. The value is a floating point number that represents the number of 0->1 and 1->0 glitch transitions, that the signal makes during a period of time. You can specify the period with the *-period* option. Alternatively, you can annotate a related clock using the *-base_clock* option, and the specified glitch rate is relative to the related clock period. Note: If either of the *-toggle_rate* or the *-glitch_rate* option is specified, the other is assumed to be zero.

For PrimePower version D-2010.06 and earlier releases, the *toggle_rate* and *glitch_rate* options are known as *-toggle_count* and *-glitch_count* respectively. If you specify the older names, the tool generates a warning message.

`-rise_ratio` *`rise_ratio`*

Specifies the ratio of rise transitions to total transitions for the specified toggle rate and glitch rate when annotating pins that are characterized with both rise and fall internal power. The *rise_ratio* argument is a floating point number between 0.0 (all transitions are falling) and 1.0 (all transitions are rising). You need to specify a toggle rate and glitch rate to use this option. The default is 0.5.

`-period` *`period_value`*

Specifies the time period for which the number of transitions given in the *toggle_rate* and *glitch_rate* occur. The time units for this value are those specified by the main technology library. When you specify this option, the values of the specified toggle and glitch rates are divided by the given *period_value*. The resulting toggle rate and glitch rate are then annotated to the objects provided in the *set_switching_activity* command. If the *-period* option is not specified, a default *period_value* of 1.0 is assumed.

`-base_clock` *`clock`*

Specifies a clock by which the toggle and glitch rate values are referenced. When applied, the toggle and glitch rates are divided by the period of the

specified clock. When the *-base_clock* value is set to "*", the toggle and glitch rates are divided by the period of the related clock.

`-type` *`object_type_list`*

Specifies a list of object types used for implicitly selecting the objects that are annotated with the specified switching activity. The *object_type_list* argument can be a list of one or more of the following object types:

- *registers* (sequential cell outputs)
- *three_states* (tristate cell outputs)
- *inputs* (input design ports / hierarchical instance pins)
- *outputs* (output design ports / hierarchical instance pins)
- *inout* (inout design ports / hierarchical instance pins)
- *ports* (design port / hierarchical instance pins)
- *nets* (nets)
- *clock_gating_cells* (clock-gating cell output)
- *black_boxes* (black box outputs)
- *non_clock_network* (nonclock network objects)
- *memory* (memory cell outputs)

When you specify the *-type* option, the tool annotates all the objects in the current instance (current design, if no current instance is set) that satisfy the selection criteria with the specified switching activity. If you specify the *registers* type, the tool annotates both the noninverting (Q) and inverting (QN) sequential cell outputs. The annotated toggle rate and glitch rate on the inverting outputs is the same as that on the noninverting outputs. The static probability on the inverting outputs is 1.0 - *value* where *value* is the specified static probability.

If you specify the *black_boxes* type, the tool annotates the black-box cell outputs. Generated clock nets are excluded from this group.

If you specify the *non_clock_network* type, the tool annotates the activity on the nonclock network cells of the design.

If you specify the *-clock_domains* option, the tool constrains the selected objects further to fall into the specified clock domains. However, the source pins of the clock are not included.

`-hierarchy`

Use with the *-type* option to specify that the objects in all the hierarchies in the current instance that satisfy the selection criteria. If not specified, the tool annotates only the top-level objects in the current instance that satisfy the selection criteria.

For PrimePower version D-2010.06 and earlier releases, by default, the objects in all the hierarchies in the current instance are considered unless you specify the *-no_hierarchy* option; For PrimePower version E-2010.12 and later, the *-no_hierarchy* option replaces the *-hierarchy* option. For the tool to consider all the hierarchies in the current instance, specify the *-hierarchy* option.

`-verbose`

Prints the detailed information, such as the design objects that are not annotated. By default, the tool suppresses detailed information.

For PrimePower version D-2010.06 and earlier releases, verbose information was provided by default. A *-quiet* option was used to suppress the detailed information. In the E-2010.12 release, detailed information is suppressed by default; the *-quiet* option replaces the *-verbose* option. If you use the invalid *-quiet* option instead of the valid *-verbose* option, a warning message is generated.

`-force`

If "-force" option is specified, then PrimePower will force the SP and TR values over any other activity which is annotated from either file (VCD, FSDB, etc.), normal "set_switching_activity", "set_case_analysis", logic constants, propagated and default activities. The "force_annotated" pin will have the highest precedence among all other activities. This command works in average as well as time based analysis mode.

`-clock_domains` *`clock_list`*

Filters the selection to include only the objects that belong to the specified clock domains and divides the toggle and glitch rate by the period of the related clock. If an object belongs to multiple clock domains, the fastest of the clocks, is selected as the related clock. Objects which belong to no clock domain, are automatically placed in the domain of the fastest design clock.

For PrimePower version D-2010.06 and earlier releases, this option is called *-clocks*. For PrimePower version E-2010.12 and later, the *-clocks* option is *-glitch_rate*. If you use the *-clocks* option, a warning message is issued.

`object_list`

Specifies a list of nets, pins, ports, or instances in the current design on which the *static_probability*, *toggle_rate*, and *glitch_rate* switching activity values are

to be set. Use this option with the *-type* argument to specify that all the objects in the given list of instances that satisfy the selection criteria are annotated. This option can also be used with or without the *-no_hierarchy* option.

Description

Use this command to annotate design nets, ports, pins, and cells with the different kinds of switching activity. These include simple toggle rate, glitch rate, and static probability on nets, ports and pins; state and path dependent toggle rate and glitch rate on cell pins, and state dependent static probabilities on cells.

Toggle rates, glitch rates, and static probabilities are annotated by using the *-toggle_rate*, *-glitch_rate* and *-static_probability* options respectively. The toggle rate, glitch rate, and static probability can be made state-dependent by specifying the state condition with the *-state_condition* option. The toggle rate and glitch rate can be made path-dependent by specifying the path sources with the *-path_sources* option. You can specify a related clock using the *-base_clock* option. The annotated toggle rate and glitch rate are divided by 1 time unit defined in the main library unless the -period, *-base_clock* or *-clock_domains* options are applied. When you specify the *-period* option, the toggle and glitch rates are divided by the period value. When you specify *-base_clock* option, the toggle and glitch rates are divided by the clock period of the specified base clock. When you specify the *-clock_domains* option, the glitch and toggle rates are divided by the period of the related clock for each domain.

The design objects that are annotated with switching activity can be specified explicitly as a list of objects. The objects can also be specified implicitly by using the *-type*, and *object_list* options together.

For statistics on the switching activity annotation on the current design, use the *report_switching_activity* command.

Examples

In the following example, tool annotates a simple toggle rate value of 33 / 1320 = 0.025, glitch rate value of 10 / 1320 = 0.0075 and a static probability value of 0.015 to all design input ports.

```
pt_shell> set_switching_activity -toggle_rate 33 -glitch_rate 10 \\
           -period 1320 -static_probability 0.015 [all_inputs]
```

The following example annotates the same activity to all input ports, relative to a clock with a period of 10.

pt_shell> *create_clock CLK -period 10* pt_shell> *set_switching_activity -toggle_rate .25 -glitch_rate .05* \\ *-base_clock CLK -static_probability .015 -type inputs*

The toggle rate value of 0.25 and glitch rate value of 0.05 are divided by clock period (10). The resulting toggle rate value of 0.025 and glitch rate value of 0.005 are annotated on the design objects.

In the following example, the tool annotates the state_condition dependent static probabilities on the cell or1:

```
pt_shell> set_switching_activity -static_probability 0.10 \\
           -state_condition "A & B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.25 \\
           -state_condition "A & ! B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.25 \\
           -state_condition "! A & B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.30 \\
           -state_condition "! A & ! B" [get_cell or1]

pt_shell> set_switching_activity -static_probability 0.10 \\
           -state_condition "default" [get_cell or1]
```

In the following example, the tool annotates simple and path dependent toggle rates and glitch rates on the output pin Y of the cell xor1:

```
pt_shell> set_switching_activity -toggle_rate 0.022 -glitch_rate 0.001 \\
           [get_pin xor1/Y]

pt_shell> set_switching_activity -toggle_rate 0.020 -glitch_rate 0.001 \\
           -path_sources "A" -state_condition "B" [get_pin xor1/Y]

pt_shell> set_switching_activity -toggle_rate 0.002 -glitch_rate 0.0001
 \\
           -path_sources "B" -state_condition "A" [get_pin xor1/Y]

pt_shell> set_switching_activity -toggle_rate 0.002 -glitch_rate 0.0001
 \\
           -path_sources "A B" -state_condition "default" [get_pin
 xor1/Y]
```

In the following example, the command annotates a toggle rate which is a fraction of the related clock on all the clock-gating cells.

```
pt_shell> set_switching_activity -clock_derate 0.1 \\
           -clock_domains [all_clocks] \\
           -type clock_gating_cells
```

The clock-gating cell outputs are annotated with a toggle rate which is 0.1 of the input clock toggle rate. If the input clock period is 10ns, its toggle rate is 2 toggle/10ns = 0.2 and the toggle rate on the clock gating output is 0.1 * 0.2 = 0.02.

In the following example, the "force" option is used. The command annotates a static probability of 1.0 and toggle rate of 0 on the A2 pin of cell "and_gate". The force value of 1 should match the value of static probability in the command.

Feedback

```
pt_shell> set_switching_activity -force 1 \\
            -toggle_rate 0 -static_probability 1.0  [get_pin and_gate/A2]
```

See Also

- create_clock
- get_switching_activity
- read_saif
- report_power
- report_switching_activity
- reset_switching_activity

---

set_synth_options

Specify RTL Architect commands to be executed before compile.

Syntax

int *set_synth_options*

```
[-file file_name]
[-cmds cmd_list]
[-synth_mode mode_name]
```

Data Types

```
file_name       string
cmd_list        list
mode_name       string
```

Arguments

`-file file_name`

Specifies tcl file which contains tcl commands to be executed before compile.

`-cmds cmd_list`

if number of tcl commands is small, specify commands directly instead of creating separate file.

`-synth_mode mode_name`

Use the -synth_mode option for logical and physical mode of synthesis. Logical mode: Used for quick clock gating analysis and lint checks. Physical mode: Used for full timing and physical aware synthesis along with the clock gating analysis and lint checks. The default is physical mode.

Feedback

Description

The *set_synth_options* command pass RTL Architect tcl commands to be executed before compile. Use this command for most of RTL Architect options which are related with SDC, Timing, Clock Gating, MultiBit, Scenario and DFT.

Examples

The following example shows how to pass RTL Architect setting.

```
pwr_shell> set_synth_options -file tz_setup.tcl
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_phy_tech_file
- set_synth_output_dir
- set_rtl_architect_shell
- set_presynth_options
- set_postsynth_options

---

set_synth_output_dir

Specify output directory.

Syntax

int *set_synth_output_dir*

```
        dir_name
```

Data Types

```
dir_name        string
```

Arguments

dir_name

Specifies output directory for all temporary files and result files. Usually it is relative path to current working directory.

Feedback

Description

The *set_synth_output_dir* command specifies output directory for all temporary files and result files.

Examples

The following example shows how to specify output directory

```
pwr_shell> set_synth_output_dir PP_WORKSPACE
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_phy_tech_file
- set_rtl_architect_shell
- set_presynth_options
- set_synth_options
- set_postsynth_options

---

set_synth_phy_tech_file

Specify LEF technology file for NDM design database

Syntax

int *set_synth_phy_tech_file*

```
[-ndm]
file_name
```

Data Types

```
file_name       string
```

Arguments

file_name

Specifies the name of the LEF technology file. if *-ndm* is specified, user can specify NDM library for technology setting.

Feedback

`-ndm`

Given file format is NDM library.

Description

The *set_synth_phy_tech_file* command pass LEF technology file to synthesis module. if *-ndm* is specified, PrimePower RTL will use technology settings from given NDM library.

Examples

The following example shows how to setup LEF techlogy file

```
pwr_shell> set_synth_phy_tech_file full_path_to/techfile.tech
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_output_dir
- set_rtl_architect_shell
- set_presynth_options
- set_synth_options
- set_postsynth_options

---

set_synth_ref_libs

Specify NDM reference libraries for synthesis

Syntax

int *set_synth_ref_libs*

```
[-power_libs file_list]
file_name ...
```

Data Types

```
file_list       list
file_name       string
```

Feedback

Arguments

file_name

Specifies the name of the NDM reference library for synthesis. PrimePower RTL honor *search_path* to find actual location of file.

-power_libs *file_list*

Specifies list of db files for power analysis in PrimePower RTL PrimePower RTL honor *search_path* to find actual location of file.

Description

The *set_synth_ref_libs* command collect NDM reference libraries to synthesis module. Use *-power_libs* option if user want to use different libraries for power analysis purpose. These libraries have higher priority than NDM reference libraries during design linking.

You can automatically generate fusion libraries for RTL synthesis. To build a Fusion Library, use the set_synth_ref_libs command with the following options:

Option Description

-lc_shell Path to the Library Compiler shell (mandatory)

-lc_tech_file To specify tech file used for Library Compiler (optional)

-lc_lef_files To specify LEF files used for Library Compiler (optional)

Examples

The following example shows how to setup NDM reference libraries

```
pwr_shell> set_synth_ref_libs A_ff.ndm B_ff.ndm

The following example shows how to specify db files for power analysis.

pwr_shell> set_synth_ref_libs -power_libs { a.db b.db } a_full.ndm
 b_full.ndm

The following example generates a complete fusion library using tech file
 and LEF files.

pwr_shell> set_synth_ref_libs -lc_shell"abc/lc_shell" -lc_tech_file
 "ts16nc.tf" -lc_lef "LEF/*.lef" ts16ncfslogl16hdl090f_ffg0p88v125c.db
 ts16ncfslogl16hdl090f_ffgnp0p88v0c.db

The following example generates a physical-less Fusion Library without
 LEF files:

pwr_shell> set_synth_ref_libs -lc_shell"abc/lc_shell"
 ts16ncfslogl16hdl090f_ffg0p88v125c.db
 ts16ncfslogl16hdl090f_ffgnp0p88v0c.db
```

```
The following example shows how to specify pre-built fusion libraries as
reference libraries using the set_synth_ref_libs command

pwr_shell> set_synth_ref_libs flibs/ts16ncfslogl16hdl090f_c
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_phy_tech_file
- set_synth_output_dir
- set_rtl_architect_shell
- set_presynth_options
- set_synth_options
- set_postsynth_options

---

set_synth_rtl_files

Specify RTL files for synthesis

Syntax

int *set_synth_rtl_files*

```
-top_module module_name
[-list_file file_name]
[-format lformat]
[-top_rtl file_name]
file_name ...
```

Data Types

```
module_name     string
file_name       string
lformat         string
```

Arguments

file_name

Specifies the name of the Verilog/SystemVerilog RTL files for synthesis.

-top_module *module_name*

Specifies top module name.

`-list_file` *`file_name`*

Specifies optional text file contains RTL file names. PrimePower RTL supports two type of list file. For *plain* format file, it contains single file name per each line. For *vcs* format file, it supports VCS list file format. Refer VCS Users manual for detail.

`-format` *`lformat`*

Format of list file. *plain* or *vcs*. Default is *plain*. For VHDL RTL file, use vcs format file list to specify these files.

`-top_rtl` *`file_name`*

Specifies top level RTL file. It is required option for vcs format of list file.

Description

The *set_synth_rtl_files* command collect RTL file related information to synthesis module especially analyze stage. Use vcs format of list file which provides most flexible and powerful features.

Examples

The following example shows how to setup two verilog files.

```
pwr_shell> set_synth_rtl_files -top_module top top.v alu.v
```

```
The following example shows how to specify vcs file list file.
```

```
pwr_shell> set_synth_rtl_files -top_module top -format vcs -list_file
 manifest.f -top_rtl full_path_to/top.v
```

See Also

- compute_metrics
- set_synth_rtl_files
- set_synth_ref_libs
- set_synth_phy_tech_file
- set_synth_output_dir
- set_rtl_architect_shell
- set_presynth_options
- set_synth_options
- set_postsynth_options

Feedback

set_target_library_subset

Sets target library subset specifications on the top-level design or hierarchical cells.

Syntax

status *set_target_library_subset*

```
[-objects objects]
[-top]
-dont_use lib_cells
```

Data Types

objects	list or collection
lib_cells	list or collection

Arguments

`-objects objects`

Specifies the hierarchical cells on which to apply the *dont_use* restrictions.

The restrictions also implicitly apply to the cell's children, if they are not explicitly assigned their own restrictions.

If you do not specify this option or the -top option, the tool uses the -top option by default.

`-top`

Sets the target library subset on the top block. The subset also applies to all cells in the hierarchy, if they are not explicitly assigned their own target library subsets.

If you do not specify this option or the -objects option, the tool uses the -top option by default.

`-dont_use lib_cells`

Specifies the library cells that cannot be used for optimization within the specified objects. This is a required option.

The library cells must exist in the reference library list. You can specify library cells by name, such as "mylib/AN2", or by collections, such as "get_lib_cells */AN2". You can also use wildcards. Wildcards are expanded according to the library cells in memory at the time the command is issued.

Description

This command sets the target library subset on the top block and hierarchical cells. Target library subsets restrict optimization on the specified objects to a subset of library cells.

Feedback

Optimization is constrained by the target library subsets on the top block and hierarchical cells. During optimization, a library cell is selected from the reference libraries. The *set_target_library_subset* command allows you to specify or filter the library cells considered within the specified objects. The subset restriction applies only to cells that are created or modified during optimization. The subset does not affect existing, unoptimized portions of the block.

A subset applied at a lower level completely overrides a subset applied at a higher level.

Use the *remove_target_library_subset* command to remove the target library subset from a top block or hierarchical cell.

Use the *report_target_library_subset* command to report the target library subset of a top block or hierarchical cell.

Examples

The following example sets a target library subset on the top-level design.

```
prompt> set_target_library_subset -top \\
   -dont_use  [get_lib_cells lib2/AND*]
```

The following example sets a target library subset on a hierarchical cell.

```
prompt> set_target_library_subset -objects [get_cells top/mid] \\
   -dont_use  [get_lib_cells lib2/INV*]
```

See Also

- report_target_library_subset
- remove_target_library_subset

---

set_temperature

Applies an operating temperature on a list of cell objects.

Syntax

int *set_temperature*

```
max_case_temperature
[-min min_case_temperature]
-object_list list_of_cells
```

Data Types

```
max_case_temperature      float
min_case_temperature      float
list_of_cells             list
```

Feedback

Arguments

max_case_temperature

Specifies the operating temperature for the maximum (worst) case.

-min *min_case_temperature*

Specifies the operating temperature for the minimum (best) case.

-object_list *list_of_cells*

Specifies the list of cells that have the operating temperatures as specified in this command.

Description

Defines the operating temperature on the cells so that these cells of the design are timed and optimized at the specified temperature. If you do not specify any operating temperature for the cells, they continue to be timed and optimized based on the available operating condition settings.

If *temperature_ranges* are specified for a cell, the operating temperatures of the cell must fall within one of the ranges specified in the *temperature_ranges*. Furthermore, *min_case_temperature* must fall within the same range as the *max_case_temperature*.

The operating temperature of a cell, once defined, can only be overridden with the *set_temperature* command used on a cell. It can also be cleared using the *reset_design* command.

Examples

The following example shows a typical context for using the *set_temperature* command, and has the max_case_temperature set at 125 on cell I1:

```
pt_shell> set_temperature 125 \\
           -object_list [get_cells I1]
1
```

The following example sets a max_case_temperature of 125 and min_case_temperature of 25 on cell I2.

```
pt_shell> set_temperature 125 \\
         -min 25 \\
         -object_list [get_cells I2]
1
```

See Also

- set_voltage

Feedback

set_timing_budget

Specifies the budget allocation mode, and provides detailed controls

Syntax

status *set_timing_budget*

```
[-mode pin_slack | clock_period]
[-slack_margin margin_value]
[-positive_slack_margin margin_value_for_positive_slack]
[-target_slack target_slack_value]
[-delay_value segment_delay_value]
[-percent_value percent_value]
[-max]
[-min]
[-rise]
[-fall]
[object_list]
```

Data Types

margin_value	float
margin_value_for_positive_slack	float
target_slack_value	float
segment_delay_value	float
percent_value	float
object_list	list

Arguments

`-mode pin_slack | clock_period`

Specifies whether the budget computation is based on the slack or the clock cycle time of the critical path through each hierarchical pin that has context captured. Please note that this option is a global setting and cannot be used with any other options that require objects, including *-delay_value*, *-percent_value*, *-max*, *-min*, *-rise*, *-fall*, *object_list*.

`-slack_margin`

Specifies a margin for negative slack paths under pin_slack mode. Please note that this option is only valid for pin_slack mode, and it can't be used together with *object_list* options.

`-positive_slack_margin`

Specifies a margin for postive slack paths under pin_slack mode. This option can be used only with the *-slack_margin* options.

`-target_slack`

Specifies a target slack value for a path segment. Please note that this option will only work for feed-through segments.

`-delay_value`

Specifies a pre-defined value as the budget of given blocks or pins. This option can be used only with the *object_list* options.

`-percent_value`

Specifies a pre-defined percentage of the clock time as the budget of given blocks or pins. Please note that this option is only valid for clock_period mode. And this option can be used only with the *object_list* options.

`-max`

Applies the configurations only when calculating budget on timing paths with max delay constraint. This option can be used only with the *object_list* options.

`-min`

Same as the *-max* option, except that the paths must be with min delay constraint. This option can be used only with the *object_list* options.

`-rise`

Applies the configurations only when calculating budget on rising timing paths. This option can be used only with the *object_list* options.

`-fall`

Applies the configurations only when calculating budget on rising timing paths. This option can be used only with the *object_list* options.

`object_list`

Specifies a list of block instances or hierarchical pins affected by the command. By default, the command modifies the context of all blocks and pins for which the context is being generated. This option can't be used with the *-mode* option.

Description

This command is used to either specify the global mode of budget allocation or to provides detailed controls for budget computation of certain segments. 1. *-mode* is to globally control the way of budget computation - pin_slack or clock_period based, while *-slack_margin* and *-positivere_slack_margin* are used to tighten or relax the slack specifically when pin_slack mode is specified.

1. Fine control for block segments are achieved through options *-delay_value*, *-percent_value* and *-target_slack*. When *-delay_value* or *-percent_value* is specified, the budget of related segments will be fixed to the value provided. When *-target_slack*

Feedback

is specified, the target slack of related segments will be fixed to the provided value, and their corresponding budgets will also be shifted accordingly. Segments with either option specified won't be considered for automatic allocation anymore. When multiple options are specified on the same segment, their priority order is *-delay_value* > *-percent_value* > *-target_slack*

2. Please note that for min delay type timing paths, only pin_slack mode can be used.

Examples

The following example specifies pin_slack mode for all blocks.

```
pt_shell> set_timing_budget -mode pin_slack
```

The following example specifies the pin_slack mode, and specifies a margin value for paths with negative slacks in the meantime.

```
pt_shell> set_timing_budget -mode pin_slack -slack_margin 1
```

The following example is same with above example, except that it also specifies a target slack value for all feed-through segments.

```
pt_shell> set_timing_budget -mode pin_slack -slack_margin 1 -target_slack
 -2
```

The following example specifies a target slack for a given block.

```
pt_shell> set_timing_budget -mode pin_slack
pt_shell> set_timing_budget -target_salck 1 blk1
```

The following example specifies a fixed budget value for path segments begin with blk1/i1.

```
pt_shell> set_timing_budget -mode pin_slack
pt_shell> set_timing_budget -delay_value 10 blk1/i1
```

See Also

- update_budget
- report_budget
- set_context_margin

---

set_timing_derate

Adjusts the calculated cell and net delays by a specified factor to add margin for on-chip variation effects.

Syntax

integer *set_timing_derate*

```
-early | -late
[-rise] [-fall]
[-data] [-clock]
[-cell_delay] [-net_delay] [-cell_check]
[-static] [-dynamic]
[-aocvm_guardband]
[-pocvm_guardband]
[-pocvm_subtract_sigma_factor_from_nominal]
[-pocvm_coefficient_scale_factor]
[-increment]
[-min_period]
[-min_pulse_width]
[-sms_scenarios sms_scenarios_list]
derate_value
[object_list]
```

Data Types

derate_value	float
object_list	list
sms_scenarios_list	collection

Arguments

`-early`

Applies the derating to the early delays (shortest paths), such as cell and net delays in the data path of a hold check.

The command must specify either *-early* or *-late*. To set both early and late derating, use two separate *set_timing_derate* commands.

`-late`

Applies the derating to the late delays (longest paths), such as cell and net delays in the data path of a setup check.

`-rise`

Applies the derating only to cell and net delays that end with a rising transition.

By default, both rising and falling delays are derated. To specify different derating values for rising and falling delays, use two separate *set_timing_derate* commands.

`-fall`

Applies the derating only to cell and net delays that end with a falling transition.

`-data`

Applies the derating only to delays in data paths. By default, delays are derated in both data paths and clock paths. To specify different derating values for data paths and clock paths, use two separate *set_timing_derate* commands.

`-clock`

Applies the derating only to delays in clock paths.

`-cell_delay`

Applies the derating only to cell delays. By default, derating applies to both cell and net delays, but not to cell timing check constraints.

`-net_delay`

Applies the derating only to net delays.

`-cell_check`

Applies the derating to cell timing check constraints instead of delays. Derating set with the *-early* option applies to hold and removal timing checks. Derating set with the *-late* option applies to setup and recovery timing checks. The *-cell_check* option cannot be used with the *-clock*, *-data*, *-cell_delay*, *-net_delay*, or *-aocvm_guardband* options.

`-static`

Applies the derating to only the static component of net delay, which is the net delay without considering crosstalk. This option works only with the *-net_delay* option. By default, both the static and dynamic delay components are derated.

`-dynamic`

Applies the derating to only the dynamic component of net delay, which is the change in delay (delta delay) caused by crosstalk. This option works only with the *-net_delay* option and cannot be used with the *-aocvm_guardband* or *-pocvm_guardband* option.

`-aocvm_guardband`

Applies the derating on top of advanced on-chip variation (AOCV) derating. The total derating applied to a timing arc is the product of the AOCV derating factor and the AOCV guardband factor.

This option has an effect only when the *timing_aocvm_enable_analysis* variable is set to *true*, and only affects the timing arcs derated by AOCV analysis. The *-aocvm_guardband* option cannot be used with the *-dynamic* or *-cell_check* option.

`-pocvm_guardband`

Applies the derating on top of parametric on-chip variation (POCV) analysis. The specified derating factor applies to both the mean and standard deviation of the statistical delay distribution.

This option has an effect only when the *timing_pocvm_enable_analysis* variable is set to *true*, and only affects the timing arcs analyzed by POCV analysis. The *-pocvm_guardband* option cannot be used with the *-dynamic* option.

`-pocvm_subtract_sigma_factor_from_nominal`

In POCV analysis, modifies the nominal value of constraint arcs of the cell instances or library cells specified by the *object_list* argument. This option can be used only with the *-cell_check* option, which modifies cell timing check constraints.

This option be applied prior to other "cell check" derating factors, such as *-pocvm_coefficient_scale_factor* and *-pocvm_guardband*.

When you use the *-pocvm_subtract_sigma_factor_from_nominal* option, the command multiplies the *derate_value* argument by the intrinsic (i.e. LVF) standard deviation of the constraint arc distribution, then subtracts the result from the existing nominal value to obtain a new nominal value.

When the library vendor has already added the standard deviation (or some fraction or multiple of it) to the nominal value, you can use this option to restore the original nominal value.

This option modifies the nominal value of the constraint arc distribution by using the standard deviation but does not affect the standard deviation itself. To modify the standard deviation, use the *-pocvm_coefficient_scale_factor* option.

`-pocvm_coefficient_scale_factor`

Applies the derating to the variation coefficient C of the statistical delay distribution in POCV analysis. This effectively scales the sigma without affecting the mean. This option has an effect only when the *timing_pocvm_enable_analysis* variable is set to *true*, and only affects the sigma of timing arcs analyzed by POCV analysis. The *-pocvm_coefficieint_scale_factor* option cannot be used with the *-dynamic* option.

To report the settings for AOCV guardband, POCV guardband, or POCV coefficient scaling, use the *report_timing_derate* command with the *-aocvm_guardband*, *-pocvm_guardband*, or *-pocvm_coefficient_scale_factor* option.

`-increment`

Applies an incremental derate setting that is added to the baseline derate settings.

Incremental derates allow you to adjust derate settings without completely replacing them. For details, see "Incremental Derate Values" below.

The *-increment* option cannot be used with AOCV/POCV options (*-aocvm_guardband*, *-pocvm_guardband*, and *-pocvm_coefficient_scale_factor*) in the same *set_timing_derate* command.

The incremental derate can be used to derate the mean delay in POCV and other derate flows.

`-min_period`

Applies the derating to cell's minimum period constraint timing check. The *-min_period* option can be used only with *-rise* or *-fall* options. *-rise* derate is used for derating rising-edge to rising-edge period, and *-fall* derate is used for derating falling-edge to falling-edge period.

To report the minimum period derating settings, use the *report_timing_derate* command with the *-min_period* option. To see the impact of derating use *report_min_period* command with *-derate* option for derate numbers.

`-min_pulse_width`

Applies the derating to cell's minimum pulse width constraint timing check. The *-min_pulse_width* option can be used only with *-rise* or *-fall* options. Derate specified with *-rise* is for derating high pulse width and *-fall* derate is used for derating low pulse width.

To report the minimum pulse width derating settings, use the *report_timing_derate* command with the *-min_pulse_width* option. To see the impact of derating use *report_min_period* command with *-derate* option for derate numbers.

`derate_value`

Specifies the derating value. This value is multiplied by the calculated cell and net delays (or timing constraint values if the *-cell_check* option is used), or added to the existing derating factor if the *-increment* option is used.

`object_list`

Specifies a list or collection of cells, library cells, or nets. The derating factor applies only to the specified objects. By default, the derating factor applies to all cell and net delays in the current design.

If settings applied to different object types overlap in scope, the setting with the narrowest object scope takes precedence. For details, see "Conflicting Derate Settings" below.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this timing derate is applied. When this option is not given the timing derate applies for all SMS scenarios.

This option is used for SMVA or SMC analysis and is currently only supported for cells and nets. This collection is created by *get_sms_scenarios*.

Description

The *set_timing_derate* command modifies calculated delays, allowing you to model on-chip variation effects. The effect is to multiply the calculated early and late delays by factors that you specify, which changes the delay and slack values reported by the *report_timing* command and other reporting commands.

The *set_timing_derate* command specifies the early or late delay adjustment factor and optionally the scope of the design affected by derating. For example,

```
pt_shell> set_timing_derate -early 0.9
pt_shell> set_timing_derate -late 1.2
```

The first command decreases all early (shortest-path) cell and net delays by 10 percent, such as those in the data path of a hold check. The second command increases late (longest-path) cell and net delays by 20 percent, such as those in the data path of a setup check. These two adjustments result in a more conservative analysis.

Using the *set_timing_derate* command implicitly changes the tool to on-chip variation mode, if not already in that mode, like using the following command:

pt_shell> *set_operating_conditions -analysis_type on_chip_variation*

In on-chip variation mode, the tool applies worst-case delay adjustments (both early and late) at the same time. For example, for a setup check, it applies late derating on the launch clock path and data path, and early derating on the capture clock path. For a hold check, it does the opposite.

In the *set_timing_derate* command, you must use either *-early* or *-late* to specify either shortest-path or longest-path delays for applying the derating. To set both early and late derating, use two separate *set_timing_derate* commands.

You must also specify the derating factor, which is a floating-point number. For a more conservative analysis that adds margin for on-chip variation effects, use a derating factor less than 1.0 for early derating or greater than 1.0 for late derating.

To report derating that has been set, use the *report_timing_derate* command.

To report the derating factor used for each incremental delay in a timing report, use the *-derate* option in the *report_timing* command:

```
pt_shell> report_timing -derate
 ...

 Path Type: max

 Point                                          Derate    Incr       Path
 ---------------------------------------------------------------------------
```

```
clock PCI_CLK (rise edge)                                  0.000     0.000
clock network delay (propagated)                           1.802     1.802
I_ORCA_TOP/I_PCI_CORE/pad_en_reg/CP (sdcrq1)               0.000     1.802 r
I_ORCA_TOP/I_PCI_CORE/pad_en_reg/Q (sdcrq1)      1.200     0.485 &   2.287 r
U7/I (inv0d1)                                    1.200     0.010 &   2.297 r
U7/ZN (inv0d1)                                   1.100     0.063 &   2.360 f
U62/I (inv0d1)                                   1.100     0.005 &   2.365 f
U62/ZN (inv0d1)                                  1.200     0.046 &   2.410 r
U63/A2 (or02d7)                                  1.200     0.005 &   2.415 r
U63/Z (or02d7)                                   1.200     0.318 &   2.733 r
...
```

To remove all derating, use the *reset_timing_derate* command.

Derating Options

By default, the *set_timing_derate* command applies the specified early (or late) derating factor to all cell and net delays in early (or late) timing path segments in the design. You can optionally modify the scope of the command in several ways:

- To derate only clock paths or only data paths, use the *-clock* or *-data* option.
- To derate only net delays or only cell delays, use the *-net_delay* or *-cell_delay* option.
- To derate only rising-edge delays, use the *-rise* option. Then the derating applies only to cell and net delays that end with a rising transition. Similarly, to derate only falling-edge delays, use the *-fall* option.
- To derate only certain nets, cell instances, or library cells, enter an object list in the command. You can use an embedded *get_nets*, *get_cells*, or *get_lib_cells* command to generate the list.
- To derate only the static or dynamic component of net delay, use the *-net_delay* option together with the *-static* or *-dynamic* option. By default, both the static and dynamic components are derated. The static component is the net delay without considering crosstalk. The dynamic component is the change in net delay (delta delay) caused by crosstalk.
- To derate cell timing check constraints instead of delays, use the *-cell_check* option. Derating set with the *-early* option applies to hold and removal timing checks. Derating set with the *-late* option applies to setup and recovery timing checks. A *derate_value* greater than 1.0 increases the value of the timing constraint, resulting in a more conservative check. By default, cell timing checks are not derated.
- To apply derating to advanced on-chip variation (AOCV) analysis or parametric on-chip variation (POCV) analysis, use the *-aocvm_guardband*, *-pocvm_guardband*, or *-pocvm_coefficient_scale_factor* option. By default, the *set_timing_derate* command affects ordinary cell and net delays, not AOCV or POCV analysis.

To report the derating options that have been set, use the *report_timing_derate* command. For example,

```
pt_shell> report_timing_derate -significant_digits 2
 ...
                          ----- Clock -----             ----- Data -----
                        Rise          Fall           Rise
 Fall
                     Early  Late    Early  Late    Early  Late    Early
 Late
-------------------------------------------------------------------------
-------
design: MYDESIGN
  Net delay static    0.90   1.20    0.90   1.10    0.90   1.20    0.90
 1.10
  Net delay dynamic   0.90   1.20    0.90   1.10    0.90   1.20    0.90
 1.10
  Cell delay          0.90   1.20    0.90   1.10    0.90   1.20    0.90
 1.10
  Cell check           --     --      --     --      --     --      --
  --

net: net198
  Net delay static    0.90   1.30    0.90   1.30    0.90   1.30    0.90
 1.30
  Net delay dynamic   0.90   1.30    0.90   1.30    0.90   1.30    0.90
 1.30
```

If you set derating on a net or leaf-level cell, the command implicitly matches the derating type to the object type: net derating for a net, or cell derating for a leaf-level cell. Derating set on a net applies to the whole net, even as it crosses into a lower level of hierarchy.

To set derating on a hierarchical cell, you must explicitly specify the type of derating by using the *-net_delay* or *-cell_delay* option (or both), or the *-cell_check* option.

If the *timing_use_constraint_derates_for_pulse_checks* variable is set to *true*, then a legacy derating method is used for *min_period* and *min_pulse_width* checks. For details, see the variable man page.

Conflicting Derating Settings

A new *set_timing_derate* command overrides any derating setting previously applied to that object.

If settings applied to different object types overlap in scope, the setting with the narrowest object scope (not the worst value) takes precedence:

- Leaf cell or net derate
- Library cell derate

- Derate applied to lowest-level enclosing hierarchical cell
- Global derate

In the following script example, the earlier commands are more specific in scope than the later ones, so the earlier ones have priority.

```
# Derate a collection of cell instances
set_timing_derate -late -cell_delay 1.4 [get_cells U2*]
#
# Derate a collection of library cells
set_timing_derate -late -cell_delay 1.3 [get_lib_cells libAZ/ND*]
#
# Derate all cell delays
set_timing_derate -late -cell_delay 1.2
#
# Derate the design (all cell and net delays)
set_timing_derate -late -1.1
```

When scaling libraries are used, and you set derating on a library cell object, the tool uses the derating value set on the library cell in the link library. To avoid ambiguity or conflict, it is recommended that you set the same derating values on the same library cell in all libraries that belong to the scaling library group.

Incremental Derate Values

To adjust derate settings without completely replacing them, two independent components of derate can be applied:

- Baseline derate
  - Specified by *set_timing_derate* or computed by the tool (such as in POCV analysis)
  - Defaults to *1.0*
- Incremental derate
  - Specified by *set_timing_derate -increment*
  - Defaults to *0.0*

When the tool computes the final derate for analysis, the incremental value is added to (not multiplied by) the baseline derate value.

Baseline and incremental derate settings can be independently applied to multiple object scopes (cells, library cells, or nets) and conditions (*-cell_delay*, *-net_delay*, *-rise*, *-fall*, and so on).

Object scope precedence is applied independently to baseline and incremental derate settings. The winning baseline derate is determined, then the winning incremental derate is determined, then both winning values are summed to compute the final derate.

To report incremental derating settings, use the *report_timing_derate* command with the *-increment* option. To see the composite (combined) derating values applied to individual cell and net delays, use the *report_timing* command with the *-derate* option.

Derating Negative Delays

Delays can be negative in some unusual situations. For example, if a cell's input transition is slow and its output transition is fast, and if the input-to-output delay is very short, the output signal can reach the 50 percent trip point before the input signal, resulting in a negative delay for the cell. A similar situation can occur for a change in net delay caused by crosstalk.

In general, delays are adjusted according to the following formula:

```
delay_new = old_delay + ( (derating_factor - 1.0) * abs(old_delay) )
```

When the delay is a positive value (the usual case), this equation is reduced to:

```
delay_new = old_delay * derating_factor
```

For negative delay, the delay adjustment equation is reduced to:

```
delay_new = old_delay * ( 2.0 - derating_factor )
```

When both the static and dynamic components of a net delay are derated, the two components are derated separately first, and then combined, so that a negative delta delay is properly derated before it is combined with a positive static delay.

Clock to data paths

If a signal is launched from a clock source, it gets a clock derate. I.e. clock to data paths are derated with clock derates.

Examples

The following commands set an early (shortest path) derating factor of 0.70 and a late (longest path) derating factor of 1.05 globally on all cell and net delays the design:

```
pt_shell> set_timing_derate -early 0.70
pt_shell> set_timing_derate -late 1.05
```

The following command sets an early derating factor of 0.75 on a specific net, n123, overriding the global derating factor for net delays:

```
pt_shell> set_timing_derate -net_delay -early 0.75 [get_nets n123]
```

The following command sets an early derating factor of 0.80 for the cell delay of cell instance U1, overriding the global derating factor for cell delays:

```
pt_shell> set_timing_derate -early 0.80 -cell_delay [get_cells U1]
```

The first command below sets a late derating factor of 1.10 on all instances of library cell IV in the library MY_LIB. The next command overrides that setting for one particular instance of that cell, inv82:

```
pt_shell> set_timing_derate -late 1.10 [get_lib_cells MY_LIB/IV]
pt_shell> set_timing_derate -late 1.05 [get_cells inv82]
```

The following command sets a late derating factor on all cells and nets in the hierarchical cell top/H1, including its lower-level hierarchical cells:

```
pt_shell> set_timing_derate -cell_delay -net_delay -late 1.05 [get_cells
 top/H1]
```

The following commands set global late and early derating factors, and then incrementally adjust those settings at particular cell instances:

```
set_timing_derate -late 1.09
set_timing_derate -early 0.91
set_timing_derate -late 1.11 [get_cells u1/u252]
set_timing_derate -increment -late 0.04 [get_cells u1/u252]
set_timing_derate -increment -early -0.03 [get_cells u1/u252]
```

These commands set a late derating factor of 1.15 (1.11 + 0.04) and an early derating factor of 0.88 (0.91 + (-0.03)) for the cell delay of cell u1/u252.

See Also

- report_design
- report_timing
- report_timing_derate
- reset_timing_derate
- set_operating_conditions
- set_voltage
- si_use_driving_cell_derate_for_delta_delay
- timing_aocvm_enable_analysis
- timing_enable_cumulative_incremental_derate
- timing_pocvm_enable_analysis

---

set_trace_option

Set an option value controlling behavior of command tracing and output annotation..

Feedback

Syntax

set_trace_option [-command name |-profile profile_metric_types |-memory_threshold threshold |-cpu_threshold threshold |-time_treshold threshold] [-annotate annnotate_type]

string *name* string *annotate_type* list rofile_metric_types *integer threshold*

Arguments

`-command` *`name`*

This option is mutually exclusive with -profile, -memory_threshold, -cpu_threshold, and -time_threshold. The option identifies the command for which annotation option is being set.

`-annotate` *`annotate_type`*

This option is meaningful only when given with the -command option. Set the annotation type of the given command with this option. The allowed values are {annotate, omit}.

The *annotate* option value causes the traced command execution to be annotated to the shell output. If the application supports an output log, the annotation is also output to the output log.

The *omit* option value causes the command execution string to be omitted from output annotation.

`-profile` *`profile_metric_types`*

This option is mutually exclusive with all others. The option sets the currently enabled profile metrics, or "all" if all metrics are enabled. Allowed values are a list of: memory, cpu, time, or the value all to indicate enabling of all metrics, or the value none to indicate disabling of all metrics. If none is given in a list with any other value, it is ignored. The enabled profile metrics are reported with command annotations in the tool output to shell, and to the output log if supported by the tool.

`-memory_threshold` *`threshold`*

This option is mutually exclusive with all others. The option sets the threshold for gathering memory metrics. The profile data annotation will be output at the end of a command inovcation only when the delta memory use for a command invocation matches or exceeds the threshold. The threshold has no effect on profile data output at the beginning of command invocation when this is enabled. The threshold is expressed in kilobyges. The special value -1 unsets the threshold.

Feedback

`-cpu_threshold` *`threshold`*

This option is mutually exclusive with all others. The option sets the threshold for gathering CPU usage metrics. The profile data annotation will be output at the end of a command inovcation only when the delta CPU usage for a command invocation matches or exceeds the threshold. The threshold has no effect on profile data output at the beginning of command invocation when this is enabled. The threshold is expressed in seconds. The special value -1 unsets the threshold. If cpu threshold isunset, then the default threshold of 1 second is used. To effect no threshold, set the threshold to 0 (zero).

`-time_threshold` *`threshold`*

This option is mutually exclusive with all others. The option sets the threshold for gathering elapsed time metrics. The profile data annotation will be output at the end of a command inovcation only when the elapsed time for a command invocation matches or exceeds the threshold. The threshold has no effect on profile data output at the beginning of command invocation when this is enabled. The threshold is expressed in seconds. The special value -1 unsets the threshold. If time threshold is unset, then the threshold for cpu is used.

Description

If a command is given with -annotate option, then the *set_trace_option* command informs the tool on how to behave during command annotation output, which is started with the *annotate_trace* command.

If profile, memory_threshold, cpu_threshold, or time_threshold is given, the the *set_trace_option* configures the behavior of performance profile gathering and the data reported in command annotations to the output.

Examples

The following example sets profile configurations, turns annotation off for the command *get_attribute*, then starts command annotations to the output.

```
prompt> set_trace_topion -profile all
prompt> set_trace_topion -memory_threshold 10
prompt> set_trace_topion -cpu_threshold 30
prompt> set_trace_topion -command get_attribute -annotate omit
prompt> annotate_trace -start -profile on
```

See Also

- log_trace
- annotate_trace
- get_trace_option

Feedback

set_units

Checks the specified units with the main library units.

Syntax

status *set_units*

```
[-time time_in_sec]
[-capacitance capacitance_in_farad]
[-current current_in_ampere]
[-voltage voltage_in_volt]
[-resistance resistance_in_ohm]
[-power power_in_watt]
```

Data Types

time_in_sec	float
capacitance_in_farad	float
current_in_ampere	float
voltage_in_volt	float
resistance_in_ohm	float
power_in_watt	float

Arguments

`-time` *time_in_sec*

Checks the time unit with the time unit of the main library.

`-capacitance` *capacitance_in_farad*

Checks the capacitance unit with the capacitance unit of the main library.

`-current` *current_in_ampere*

Checks the current unit with the current unit of the main library.

`-voltage` *voltage_in_volt*

Checks the voltage with the voltage unit of the main library.

`-resistance` *resistance_in_ohm*

Checks the resistance unit with the resistance unit of the main library.

`-power` *power_in_watt*

Checks the power unit with the leakage power unit of the main library. The power units can only be checked if power analysis mode is enabled.

Feedback

Description

The *set_units* command checks the specified units with the main library units. The command issues a warning if the specified units do not match the main library units. You can use this command only after linking the design.

The main library units are the design units. The *set_units* command does not set the design units, but it prevents the inadvertent use of the wrong units from different SDC files.

The power units can only be checked if power analysis mode is enabled. This can be done by setting the *power_enable_analysis* variable to *true* and running power related commands, such as *update_power*, *read_vcd*, *read_saif*, and *set_switching_activity*.

To report all units, run the *report_units* command. The *read_sdc* command supports the *set_units* command. The *write_sdc* and *write_script* commands write the *set_units* command as the first command at their output.

The *set_units* command does not affect the output of reporting commands. Reports continue to use the units defined in the main library.

Examples

The following example checks the if the main library time unit is 1ns, capacitance unit is 1pF, current unit is 1mA and voltage unit is 1V.

```
pt_shell> set_units -time ns -capacitance pF -current mA -voltage V
```

The following example checks if the main library capacitance unit is 0.5pF.

```
pt_shell> set_units -capacitance 0.5pF
```

See Also

- read_sdc
- report_units
- write_script
- write_sdc

---

set_unix_variable

This is a synonym for the *setenv* command.

See Also

- printenv
- printvar

Feedback

- setenv
- sh

---

set_user_attribute

Sets a user attribute to a specified value on an object.

Syntax

string *set_user_attribute*

```
[-class class_name]
[-quiet]
object_spec
attr_name
value
```

Data Types

```
class_name          string
object_spec         list
attr_name           string
value               string
```

Arguments

`-class class_name`

If the *object_spec* option is a name, this is its class. Allowable values are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, *lib_pin*, *timing_path*, *clock*, or *lib_timing_arc*.

`-quiet`

Suppresses all report messages.

object_spec

Objects on which to set the attribute. Each element in the list is a collection or a pattern which is combined with the *class_name* option to find the objects.

attr_name

Shows the name of the attribute.

value

Shows the value of the attribute.

Description

The *set_user_attribute* command sets attributes on objects. These attributes are defined with the *define_user_attribute* command.

You cannot set application attributes with this command. Each settable application attribute has a command dedicated to it. For example, the *fanout_load* attribute is set with the *set_fanout_load* command. However, you can still retrieve the values of any application or user-defined attribute using the *get_attribute* command.

Examples

This example defines an attribute 'X' for cells, then sets the value on all cells in this level of the hierarchy.

```
pt_shell> define_user_attribute -type int -class cell X
pt_shell> set_user_attribute [get_cells *] X 30
Set attribute 'X' on 'i1'
Set attribute 'X' on 'i2'
```

To be queried, the attribute must be set on a timing path collection that has been set to a variable, for example,

```
pt_shell> define_user_attribute -classes timing_path -type boolean
 is_special
pt_shell> set mypath [get_timing_paths ...]
pt_shell> set_user_attribute -class timing_path $mypath is_special true
pt_shell> get_attribute $mypath is_special
true
```

This behavior differs from user attributes set on other objects, such as pins, which allow user attributes to be set and queried directly in the design (not stored in a collection variable).

The *get_timing_paths* command constructs a new timing path at every invocation so two timing paths derived from identical get_timing_paths commands and stored in two different collection variables do not share the same user attributes. For example,

```
pt_shell> define_user_attribute comment -type string -class timing_path
pt_shell> set p1 [get_timing_paths]
pt_shell> set p2 [get_timing_paths]
pt_shell> set_user_attribute $p1 comment "Hello world"
pt_shell> get_attribute $p1 comment
Hello world
pt_shell> get_attribute $p2 comment
Warning: Attribute 'comment' does not exist on timing_path
 'path' (ATTR-3)
```

See Also

- collections
- define_user_attribute
- get_attribute

Feedback

- list_attributes
- remove_user_attribute
- report_attribute

---

set_user_units

This command sets the units that are used for user input.

Syntax

string *set_user_units*

```
[-input]
-type unit_type
-value unit_value

string unit_type
string unit_value
```

Arguments

`-input`

Set the units for data input such as source and read_sdc.

`-type unit_type`

Specifies the type of unit to set. This is one of "capacitance", "current", "length", "power", "resistance", "time" or "voltage".

`-value unit_value`

Specifies the unit value. For time, equivalent examples would include "1ps", "1.0ps", "1e-12".

Description

This command sets the units that are used for data input. The tool maintains separate units for input and input. There are unit values for time, resistance, capacitance, power, voltage and current.

Examples

The following example sets input units to 10ps, 1kOhm, 1pF, 1V and 1mA.

```
prompt> set_user_units -type time -value 10ps
prompt> set_user_units -type resistance -value 1kOhm
prompt> set_user_units -type capacitance -value 1pF
prompt> set_user_units -type voltage -value 1V
prompt> set_user_units -type current -value 1mA
```

The following example sets input time units to 1ps.

```
prompt> set_user_units -input -type time -value 1ps
```

---

set_vector_generation_options

Sets the options for vector free vector generation.

Syntax

integer *set_vector_generation_options*

```
[-enable_activity true | false]
[-propagate_zero_activity_nets true | false]
[-event_start_time event_start_time]
[-time_unit ps | ns]
[-retention_mode save | restore]
[-seed_objects object_list]
[-number_of_cycles number_of_cycles]
[-clock object_list]
[-include_ports true | false]
[-events_on_pins true | false]
[-events_on_user_specified_nets_only true | false]
[-special_nets object_list]
[-toggle_percentage toggle_percentage]
[-reset]
```

Data Types

event_start_time	integer
object_list	list
number_of_cycles	integer
toggle_percentage	float

Arguments

`-enable_activity`

Enables using switching activity and state probability to guide single cycle vector generation. Default is true. This option is not honored in multi-cycle vector generation.

`-propagate_zero_activity_nets`

Treats zero switching activity nets as logic constant nets, and propagate the logic forward and backward during vector generation. Default is true.

`-event_start_time`

Specifies start time for the first event in the VCD file. By default the first event starts at 1 time unit, where the time unit is the time unit in *report_unit* command.

`-time_unit`

Specifies time unit in the VCD file. Default is the time unit in *report_unit*. Accepted option is *ps* or *ns*.

`-retention_mode`

Specifies mode for retention cells during single cycle vector generation. Accepted option value is save or restore. Default is save mode. This option is not honored in multi-cycle vector generation.

`-seed_objects`

Specified a list of cell objects as seeds such that vectors will be generated from those seed objects first.

`-number_of_cycles`

Specifies number of clock cycles during multi-cycle vector generation. This number is based on dominant clock, which is defined as the clock which has most power associated with. Default value is 1, which means default is single cycle vector generation.

`-clock`

Specifies clock object list for multi-cycle vector generation. By default, vectors are generated for cells on all clocks. When clock object list is specified, vectors will only be generated on cells associated with those clocks in the list.

`-include_port`

Enables ports to be included in VCD or FSDB file. Default value is false, which means only RTL nets are included.

`-events_on_pins`

Enables pins to be includes in VCD or FSDB file. Default value is false, which means only RTL nets are included.

`-events_on_user_specified_nets_only`

Enables writing out events for user specified nets only to VCD or FSDB file. Default value is false, which means all RTL nets will be written out. User specified nets are net objects specified in the command *set_net_pattern*. This option is only honored in multi-cycle vector generation.

`-special_nets`

Specifies net objects to be considered in special mode vector generation. When this option is specified, special mode vector generation will be turned on. Vectors will be generated for single clock cycle only. Default is empty list, which means special mode vector generation is off.

`-toggle_percentage`

Specifies percentage of nets to be toggled during special mode vector generation. It is to be used with the option *-special_nets*. Default value is 0, which means none of the nets will be toggling. This is the only option honored in special mode vector generation, and all other options are not honored.

`-reset`

Resets all options in the command to default values.

Description

This command is used to specify options for the command *write_vectors*, while the command *write_vectors* does not take any option. Those options are specified such that different VCD files can be generated where different events are generated.

There are three different types of vectors can be generated, and each one can be specified by corresponding option in this command. Default is single clock cycle vector generation. The second one is multi-cycle vector generation, which is enabled by the option *-number_of_cycles*. The third one is special mode vector generation for single cycle, which is enabled by the option *-special_nets*.

Examples

The following example sets clock gating percentage to 20% instead of default 50%, memory percentage to 25% instead of default 50%, and then invokes the command *write_vectors*.

```
pt_shell> set_clock_gating_percentage -value 0.2
pt_shell> set_memory_percentage -value 0.25
pt_shell> write_vectors rtl.vcd
```

The following example disables using switching activity and state probability for the command *write_vectors*. At the same time it disables propagting zero activity nets. In the second invocation of the command *write_vectors*, it resets all options to default values, and generates a new VCD file.

```
pt_shell> set_vector_generation_options -enable_activity false
pt_shell> set_vector_generation_options -propagate_zero_activity_nets
 false
pt_shell> write_vectors rtl_no_activity.vcd
pt_shell> set_vector_generation_options -reset
pt_shell> write_vectors rtl_default.vcd
```

The following example specifies a list of buffers as seeds for vector generation.

```
pt_shell> set_vector_generation_options -seed_objects [get_cells
 inst_buf*]
pt_shell> write_vectors rtl_default.vcd
```

The following example shows how to generate a 100 cycle VCD file.

```
pt_shell> set_vector_generation_options -number_of_cycles 100
pt_shell> write_vectors 100_cycle.vcd
```

The following example shows how to generate a special mode VCD file, where all nets are considered as special nets, and 75% of those nets are toggling. At the same time, state 0 is set on all clock gating cell enable nets through the command *set_net_pattern*.

```
pt_shell> set_vector_generation_options -special_nets [get_nets -hier]
pt_shell> set_vector_generation_options -toggle_percentage 0.75
pt_shell> set_net_pattern -pattern 0 -object_list [get_nets -of [get_pins
 icg_*/enble]
pt_shell> write_vectors special_mode_75.vcd
```

The following example shows how to utilize the option *-events_on_user_specified_nets_only* to achieve the goal: only random events are generated on primmary ports, and all internal nets will get events propagated including RTL points during power analysis.

```
pt_shell> set_net_pattern -object_list [get_nets -of [get_ports]] -mode
 constraint
pt_shell> set_vector_generation_options -number_of_cycles 100
pt_shell> set_vector_generation_options
 -events_on_user_specified_nets_only true
pt_shell> write_vectors pi.vcd
```

See Also

- write_vectors
- set_memory_percentage
- set_clock_gating_percentage
- set_net_pattern

---

set_voltage

Specifies the operating voltage for a list of power nets or PG pins.

Syntax

int *set_voltage*

```
[-min min_case_voltage]
[-min_dynamic dynamic_min_case_value]
[-dynamic dynamic_max_case_value]
[-object_list list_of_supply_nets]
[-cell cell]
[-pg_pin_name pg_pin]
```

```
[-reference_name voltage_level_name]
[-reset]
max_case_voltage
[-nominal nominal_voltage]
```

Data Types

min_case_voltage	float
dynamic_min_case_value	float
dynamic_max_case_value	float
list_of_supply_nets	list
cell	list
pg_pin	string
max_case_voltage	float
voltage_level_name	string
reset	boolean
nominal_voltage	float

Arguments

`-min` *min_case_voltage*

Specifies the operating voltage for the minimum delay (fastest) case. This is typically larger than the *max_case_voltage* value.

`-min_dynamic` *dynamic_min_case_value*

Specifies the dynamic portion of the full (static+dynamic) min-delay rail voltage value. This is the largest dynamic voltage change, specified as a small positive delta, that can occur in min-delay (high voltage) analysis. For details, see EXAMPLES.

`-dynamic` *dynamic_max_case_value*

Specifies the dynamic portion of the full (static+dynamic) max-delay rail voltage value. This is the largest dynamic voltage change, specified as a small negative delta, that can occur in max-delay (low voltage) analysis. For details, see EXAMPLES.

`-object_list` *list_of_supply_nets*

Specifies a list of supply nets or supply groups on which to set the operating voltage.

`-cell` *cell*

Specifies a cell instance on which to apply the operating voltage for IR drop annotation.

`-pg_pin_name` *pg_pin*

Specifies the name of the PG pin of the cell on which to apply the operating voltage for IR drop annotation purposes.

The *-cell* and *-pg_pin_name* options together specify the power or ground pin of a cell instance on which to apply the operating voltage. These two options are always used together and cannot be used with the *-object_list* option.

max_case_voltage

Specifies the operating voltage for the maximum delay (slowest) case. This is typically less than the *-min min_case_voltage* value.

-reference_name *voltage_level_name*

Specifies the name of the voltage level associated with the supply group object specified with the -object_list option. The reference name is defined using a *set_voltage_levels* command. The specified voltage information is associated with this reference name and is used in SMVA analysis.

-reset

Resets the voltage of the nets specified with optional -object_list option. If object list is not specified, then the voltage of all supplied nets is reset. All other options except object_list are invalid with -reset. After reset, report_supply_net will not show any voltage for the supply nets whose voltage is reset. Please note that if voltage on supply net is reset calculation will fallback to voltage on corresponding PG pins.

-nominal *nominal_voltage*

Specifies an operating voltage for the rail. This should be a nominal value between the *max_case_voltage* value and the *min_case_voltage* value. This option cannot be used with *-cell* option.

Description

This command defines the operating voltage on power nets or PG pins so that the parts of the design powered by these nets are analyzed at the specified voltage. The specified voltage must be consistent with the voltage ranges defined by the UPF commands.

If you do not specify any operating voltage for a power net, the part of the design connected by the power net is analyzed according to the voltage defined under the *voltage_map* attribute in Liberty syntax for the library cell, or by the operating conditions set by the *set_operating_conditions* command.

After you set the operating voltage of a power net with the *set_voltage* command, you can override that setting by using the *set_voltage* command again. The *reset_design* clears all *set_voltage* settings.

When you set the voltage on UPF supply nets, the voltage is set only on the net segments existing at that time that are connected to the supplies specified by the *-object_list* option. Therefore, it is important to load the complete UPF description of the power network before you use the *set_voltage* command.

To check whether any supply net segment is missing a voltage assignment, use the *check_timing -override_defaults supply_net_voltage* command.

Examples

The following command sets the voltages for the supply net named VDD1 for maximum-delay analysis (0.89 volts) and minimum-delay analysis (0.91 volts):

```
pt_shell> set_voltage -object_list VDD1 0.89 -min 0.91
```

The following script defines the supply nets of a power domain and sets specific power and ground voltages for cell instance U18, overriding the supply voltages set by the *create_operating_conditions* and *set_operating_conditions* commands at the design level, or the default supply voltages defined in the library.

```
create_power_domain TOP
create_supply_net VDD -domain TOP
create_supply_net VSS -domain TOP
set_domain_supply_net TOP -primary_power_net VDD \\
                          -primary_ground_net VSS
set_voltage 0.88 -min 1.188 -cell U18 -pg_pin_name PWR
set_voltage 0.04 -min -0.05 -cell U18 -pg_pin_name GND
```

The first *set_voltage* command sets a supply voltage of 0.88 for maximum delay analysis and 1.188 for minimum delay analysis. The second one sets a ground voltage of 0.04 for maximum delay analysis and -0.05 for minimum delay analysis.

The following command specifies the dynamic part of the supply voltage for the supply net named VDD1:

```
pt_shell> set_voltage -object_list VDD1 \\
       0.80   -dynamic     -0.05 \\
  -min 0.90   -min_dynamic  0.02
```

In this example, the operating voltage can vary between 0.80 and 0.90. These are the total (static+dynamic) values. The specified dynamic values are the dynamic components of these total values:

```
total = static + dynamic
```

This maps to the preceding example as follows:

```
Maximum delay analysis:   0.80    =  0.85    +  (-0.05)
Minimum delay analysis    0.90    =  0.88    +  (+0.02)
                          (total) = (static) + (dynamic)
```

During clock reconvergence pessimism removal (CRPR), the tool removes the early/late delay differences leading up to the common point.

For a typical setup path (launch and capture come from different edges in time), CRPR removes the static early/late difference at the common point, but leaves the dynamic early/

late difference in place. Accordingly, CRPR computes the adjustment using static min/max supply voltages along the common path cells so that only the static early/late difference is removed. (This static-only voltage computation occurs entirely within CRPR; arrivals propagation and reporting use static+dynamic values as usual.)

For a typical hold path (launch and capture come from the same edge in time), CRPR removes the static+dynamic early/late difference because the same physical edge cannot be both early and late at the same time. In this case, CRPR computes the adjustment using the full (static+dynamic) min/max supply voltages along the common path cells, so that all of the static+dynamic early/late delay difference at the common point is removed.

See Also

- check_timing
- create_power_domain
- create_supply_net
- report_power_domain
- report_power_pin_info
- set_operating_conditions
- set_voltage_levels

---

set_voltage_levels

Assings a supply group to have multiple voltage levels for SMVA analysis.

Syntax

int *set_voltage_levels*

```
supply_group_list
[-reference_names reference_names_list]
[-reset]
```

Data Types

supply_group_list	string
reference_names_list	list

Arguments

supply_group_list

Specifies the supply groups for which the voltage levels will be set.

`-reset`

Reset the voltage levels to undefined state.

`-reference_names` *`reference_names_list`*

Specifies the reference voltage names associated with the supply groups.

Description

The *set_voltage_levels* command is used to assign multiple voltage levels to a given supply_group. These voltage levels are used in SMVA analysis. The voltage levels get created with labels 'low' and 'high'. In SMVA analysis all combinations of high and low voltage levels of the supply_groups on the path are explored. The supply_groups that don't have voltage levels assigned are treated as fixed voltage in SMVA analysis.

The voltage level assignment can be reported using the report_supply_group command.

Examples

The following example shows a typical usage of the *set_voltage_levels* command:

```
prompt> set_voltage_levels T.primary.power -reference_names { low high }
1
```

See Also

- report_supply_group
- set_voltage

---

set_vsa_margin_table

Defines the margin table consisting of derating factors against different voltages, for a library macro cell that does not have voltage scaling libraries.

Syntax

status *set_vsa_margin_table*

`-cell` *`lib_cell`*
`margin_table`

DATA TYPES

lib_cell string *margin_table* list

Arguments

`-cell` *`lib_cell`*

Specifies a marco cell instance that does not have voltage scaling libraries.

margin_table

The syntax of *margin_table* can be expressed as:

```
{ {VDD1 derating_factor1} {VDD2 derating_factor2} {VDD3
 derating_factor3} ... }
```

Description

When VSA calculates the delay of a marco cell with a set margin table, VSA looks up the table and applies the corresponding derating factor to its delay at the nominal voltage, to get the delay at a specified voltage.

When VSA computes delay at a voltage that is not specified in the margin table, VSA uses interpolation or extrapolation to obtain the derating factor.

An example of margin_table is as follows:

```
{ {0.5 1.10} {0.55 1.05} {0.6 1.0} {0.65 1.03} {0.7 1.8} }
```

When VSA computes slack at *0.55 V*, VSA uses *1.05* as the derating factor.

When VSA computes slack at *0.57 V*, VSA uses *1.02* as the derating factor which is interpolated from *{0.55 1.05}* and *{0.6 1.0}*.

Examples

The following command sets a margin table to a library cell *MACRO*, for voltage slack analysis.

```
pt_shell> set_vsa_margin_table -cell MACRO { {0.5 1.10} {0.55 1.05} {0.6
 1.0} {0.65 1.03} {0.7 1.8} }
```

See Also

- reset_vsa_margin_table

---

set_vt_mistracking_derate

Specifies the derating factors used to model Vt mistracking effects.

Syntax

integer *set_vt_mistracking_derate*

```
-min value
-max value
object_list
[-vt_type name]
```

Data Types

value	float
name	string
object_list	collection

Arguments

`-min` *value*

Specifies the min-delay Vt mistracking derating factor.

`-max` *value*

Specifies the max-delay Vt mistracking derating factor.

object_list

Specifies the collection of *lib_cell* objects to apply the specification to.

`-vt_type` *name*

Specifies a Vt type name. This is a user-defined label that allows the tool to differentiate the specifications.

Description

Vt mistracking effects are modeled by a set of min/max corner derate factors, applied to library cells with this command.

Issue this command once for each Vt class. For each command,

- Specify a user-defined Vt class name with the *-vt_type* option, such as *hvt*, *svt*, or *lvt*.
- Specify the collection of library cells belonging to this Vt class.
- Specify min-delay and max-delay derate factors that model the extent of the Vt mistracking effect.

These Vt mistracking derate specifications are a prerequisite for both bounding Vt mistracking analysis and advanced path-based Vt mistracking analysis.

In bounding Vt mistracking analysis, opposing Vt derate values are applied to the launch and capture sides of timing analysis. This is a pessimistic analysis (same-Vt cells get opposing derates across launch and capture), but it is a bounding analysis that ensures no paths are missed. See the *vt_mistracking_enable_bounding_analysis* man page for details.

In advanced Vt mistracking analysis, the min/max derating values are used to build less pessimistic mistracking models in path-based analysis (PBA):

- In enumeration mode, all min/max Vt-family configurations are explored.
- In Monte Carlo mode, statistical models are used (optionally with user-specified partial correlations).

See the *vt_mistracking_enable_path_analysis* man page for details.

Examples

The following example shows a typical usage of the *set_vt_mistracking_derate* command.

```
prompt> set_vt_mistracking_derate -min 0.95 -max 1.05 -vt_type svt
 [get_lib_cell */*svt*]
prompt> set_vt_mistracking_derate -min 0.96 -max 1.04 -vt_type lvt
 [get_lib_cell */*lvt*]
prompt> set_vt_mistracking_derate -min 0.97 -max 1.03 -vt_type ulvt
 [get_lib_cell */*ulvt*]

prompt> set vt_mistracking_enable_bounding_analysis true
prompt> update_timing -full

prompt> set vt_mistracking_enable_path_analysis true

prompt> set vt_paths \\
          [get_timing_paths \\
            -pba_mode path \\
            -path_type full_clock -nworst 10 \\
            -max_paths 2000000 -slack_lesser_than 0.0]
```

See Also

- vt_mistracking_enable_path_analysis
- vt_mistracking_enable_bounding_analysis

---

set_vt_skew_derate

Specifies the derating factors used to model Vt skew effects.

Syntax

integer *set_vt_skew_derate*

```
-min value
-max value
object_list
[-vt_type name]
```

Feedback

Data Types

value	float
name	string
object_list	collection

Arguments

`-min` *value*

Specifies the minimum-delay Vt skew derating factor.

`-max` *value*

Specifies the maximum-delay Vt skew derating factor.

object_list

Specifies the collection of *lib_cell* objects to apply the specification to.

`-vt_type` *name*

Specifies a Vt type name. This is a user-defined label that allows the tool to differentiate the specifications.

Description

Vt skew effects are modeled by a set of minimum/maximum corner derate factors, applied to library cells with this command.

Issue this command once for each Vt class. For each command,

- Specify a user-defined Vt class name with the *-vt_type* option, such as *hvt*, *svt*, or *lvt*.
- Specify the collection of library cells belonging to this Vt class.
- Specify minimum-delay and maximum-delay derate factors that model the extent of the Vt skew effect.

These Vt skew derate specifications are a prerequisite for both bounding Vt skew analysis and advanced path-based Vt skew analysis.

In bounding Vt skew analysis, opposing Vt derate values are applied to the launch and capture sides of timing analysis. This is a pessimistic analysis (same-Vt cells get opposing derates across launch and capture), but it is a bounding analysis that ensures no paths are missed. See the *vt_skew_enable_bounding_analysis* man page for details.

In advanced Vt skew analysis, the minimum/maximum derating values are used to build less pessimistic skew models in path-based analysis (PBA):

- In enumeration mode, all minimum/maximum Vt-family configurations are explored.
- In Monte Carlo mode, statistical models are used (optionally with user-specified partial correlations).

See the *vt_skew_enable_path_analysis* man page for details.

Examples

The following example shows a typical usage of the *set_vt_skew_derate* command.

```
prompt> set_vt_skew_derate -min 0.95 -max 1.05 -vt_type svt [get_lib_cell
 */*svt*]
prompt> set_vt_skew_derate -min 0.96 -max 1.04 -vt_type lvt [get_lib_cell
 */*lvt*]
prompt> set_vt_skew_derate -min 0.97 -max 1.03 -vt_type ulvt
 [get_lib_cell */*ulvt*]

prompt> set vt_skew_enable_bounding_analysis true
prompt> update_timing -full

prompt> set vt_skew_enable_path_analysis true

prompt> set vt_paths \\
          [get_timing_paths \\
            -pba_mode path \\
            -path_type full_clock -nworst 10 \\
            -max_paths 2000000 -slack_lesser_than 0.0]
```

See Also

- vt_skew_enable_path_analysis
- vt_skew_enable_bounding_analysis

---

set_waveform_integrity_analysis

Sets static or dynamic constraints for waveform integrity analysis.

Syntax

status *set_waveform_integrity_analysis*

```
[-static]
[-dynamic]
[-sms_scenarios sms_scenarios_list]
constraint_value
[object_list]
```

Data Types

constraint_value	float
object_list	list
sms_scenarios_list	collection

Feedback

Arguments

`-static`

Applies the constraint to static waveform integrity analysis.

`-dynamic`

Applies the constraint to dynamic waveform integrity analysis.

`constraint_value`

Specifies the constraint limit (Value >= 0). For static analysis, this should be in library time unit. For dynamic analysis, this should be percentage of the supply voltage.

`object_list`

Specifies a list of cell input pins.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios for which this waveform integrity information is applied. When this option is not given the waveform integrity information applies for all SMS scenarios. This collection is created by *get_sms_scenarios*.

Description

Specifies a constraint limit for waveform integrity analysis. For static analysis, the constraint value is the constraint limit for waveform distortion. If the distorted waveform from the advanced waveform propagation has larger distortion metric than the constraint value, it is reported by the *report_constraint -waveform_analysis static* command.

The distortion metric is calculated in the max time difference between the distorted waveform and its concave bounding waveform at the same voltage level. Thus, the constraint value for static analysis should be in library time units. Note that the static constraint can be set only to the input pins of synchronous elements of the design. Since it requires the waveform, advanced waveform propagation should be enabled for static waveform integrity analysis.

For dynamic analysis, the constraint value is the constraint limit for the aggressor bumps. If the sum of all active aggressors to this input pin is larger than the constraint limit, it is reported by the *report_constraint -waveform_analysis dynamic* command. Thus, the constraint value for dynamic analysis should be a percentage of the supply voltage. Note that the dynamic constraint can be set to any input pins of the design, and it is available only when signal integrity analysis is enabled.

The constraint value is applied to the specified cell input pins. If no pins are specified, the constraint value is applied as a design-level constraint. When both design-level and pin-level constraints are specified, the most restrictive constraint is used.

Feedback

Examples

The following example sets a static constraint of 0.3 library time unit on synchronous input pin DFF/CLK.

```
pt_shell> set_waveform_integrity_analysis -static 0.3 [get_pins DFF/CLK]
```

The following example sets a dynamic constraint of 0.5 on input pin mycell/A.

```
pt_shell> set_waveform_integrity_analysis -dynamic 0.5 [get_pins
 mycell/A]
```

See Also

- remove_waveform_integrity_analysis
- report_waveform_integrity_analysis
- report_constraint

---

set_wire_load_min_block_size

Sets the minimum block area for automatic wire load selection. Any blocks with an area below the minimum are promoted to the minimum.

Syntax

int *set_wire_load_min_block_size*

block_size

Data Types

block_size	float

Arguments

block_size

Minimum block size for auto wire load selection. This float value must be greater than or equal to 0.0.

Description

Specifies the minimum block area for automatic wire load selection in the current design. When automatic wire load selection is enabled, hierarchical cells are assigned wire load models based on their area. Any cells with an area below the minimum block size are assigned the minimum *block_size* value before a wire load model is selected. If the minimum block size is not explicitly set, the default value of 0.0 used.

Feedback

Examples

This example specifies a minimum block size of 1000 area units.

```
pt_shell> set_wire_load_min_block_size 1000
```

See Also

- report_wire_load
- set_wire_load_selection_group
- auto_wire_load_selection

---

set_wire_load_mode

Sets wire load mode for the current design.

Syntax

string *set_wire_load_mode*

mode_name

Data Types

mode_name	string

Arguments

mode_name

Name of mode: *top*, *enclosed*, or *segmented*.

Description

Specifies a wire load mode with the *set_wire_load_mode* command. If the mode for the top level design is *top*, the wire load model on hierarchical cells has no effect, and the top-level wire load model is used to compute wire capacitance for all nets within the design.

If the mode for the top-level design is either *enclosed* or *segmented*, wire load models on hierarchical cells are used to calculate wire capacitance, resistance, and area in nets inside these blocks. If the *enclosed* mode is set, net values are determined using the wire load model of the hierarchical cell which fully encloses the net. If the *segmented* mode is set, segments of the net in each level of hierarchy are calculated separately, and the net value is the total of all segments. Hierarchical boundary pins are included in the pin count for the segment.

If no *wire_load_model_mode* is specified for a design, a default wire load mode is searched for in the first library in the link path. If the library checked does not have a default set, *top* is assumed.

Examples

This example sets the wire load mode to *enclosed* on the current design.

```
pt_shell> set_wire_load_mode enclosed
```

See Also

- report_wire_load
- set_wire_load_model

---

set_wire_load_model

Sets wire load model on designs, ports, or hierarchical cells.

Syntax

int *set_wire_load_model*

```
-name model_name
[-library lib_spec]
[-min]
[-max]
[object_list]
```

Data Types

```
model_name        string
lib_spec          string
object_list       list
```

Arguments

`-name model_name`

Specifies the name of the wire load model; must be a valid wire load model in a library specified in the *link_path* command.

`-library lib_spec`

Specifies the library in which to search for the *model_name* variable. If not specified, libraries in the *link_path* command are searched.

`-min`

Indicates that the model is for minimum conditions.

`-max`

Indicates that the model is for maximum conditions.

`object_list`

Specifies a list of designs, ports, or hierarchical cells. The default is to set the wire model on the current instance, if set; or on the current design otherwise.

Description

Sets the wire load model on designs, ports, or hierarchical cells. The wire load model calculates net capacitance, resistance, and area for designs that have not been placed and routed.

Use automatic wire load selection for the design as another way to specify wire load models. If this is enabled, hierarchical blocks have wire load models assigned automatically based on their cell area.

To remove the wire load model setting, use the *remove_wire_load_model* command.

Examples

The following example specifies a wire load model of "BIG" on the top level design, a model of "MEDIUM" on hierarchical cell "u1", and a model of "SMALL" on hierarchical cell u1/u2.

```
pt_shell> current_design TOP
{"TOP"}
pt_shell> set_wire_load_model -name BIG
1
pt_shell> current_instance u1
u1
pt_shell> set_wire_load_model -name MEDIUM
1
pt_shell> current_instance u2
u1/u2
pt_shell> set_wire_load_model -name SMALL
1
pt_shell> current_instance
Current instance is the top-level of design 'TOP'.
pt_shell>
```

The following example specifies a wire load model of "BIG" on the top-level design, and a model of "SMALL" on all instances of design "adder."

```
pt_shell> current_design TOP
{"TOP"}
pt_shell> set_wire_load_model -name BIG
1
```

Feedback

```
pt_shell> set_wire_load_model -name SMALL [all_instances -hierarchy
 adder]
1
```

The following example sets the external wire load model for ports OUT1* to 70x70, and specifies an external fanout number of 2.

```
pt_shell> set_wire_load_model 70x70 [get_ports OUT1*]
pt_shell> set_port_fanout_number 2 [get_ports OUT1*]
```

See Also

- remove_wire_load_model
- report_lib
- report_wire_load
- set_port_fanout_number
- set_wire_load_mode
- set_wire_load_selection_group
- auto_wire_load_selection
- link_path

---

set_wire_load_selection_group

Sets the wire load selection group for current design.

Syntax

int *set_wire_load_selection_group*

```
[-min]
[-max]
[-library lib_spec]
selection_group_name
[object_list]
```

Data Types

lib_spec	string
selection_group_name	string
object_list	list

Feedback

Arguments

`-min`

Specifies the selection group for minimum conditions.

`-max`

Specifies the selection group for maximum conditions.

`-library` *`lib_spec`*

Specifies the library in which to search for the *selection_group_name* variable. If it is not specified, libraries in the *link_path* command are searched.

`selection_group_name`

Name of the selection group. This must be a valid selection group in a library specified in the *link_path* command.

`object_list`

Provides a list of hierarchical cells or designs. If you do not specify the *object_list* variable, the wire load selection group is set on the current instance, or on the current design if current instance is not set.

Description

Specifies the name of the selection group used when determining the wire load model based on cell area of blocks, when the *auto_wire_load_selection* command is true. This option is required only when the main library has more than one selection group defined, and the default group defined in the library is not the desired group. Automatic wire load selection is supported only for the *enclosed* wire load mode, specified using the *set_wire_load_mode* command.

To remove the wire load selection group setting, use the *remove_wire_load_selection_group* command.

Examples

This example specifies that the selection group named "selgrp1" from library "tech_lib" be applied to the current design.

```
pt_shell> set_wire_load_selection_group selgrp1 -library tech_lib
```

See Also

- remove_wire_load_selection_group
- report_wire_load
- report_lib

Feedback

- set_wire_load_min_block_size
- set_wire_load_mode
- set_wire_load_model
- auto_wire_load_selection
- link_path

---

setenv

Sets the value of a system environment variable.

Syntax

string *setenv variable_name new_value*

string *variable_name*

string *new_value*

Arguments

variable_name

Names of the system environment variable to set.

new_value

Specifies the new value for the system environment variable.

Description

The *setenv* command sets the specified system environment *variable_name* to the *new_value* within the application. If the variable is not defined in the environment, the environment variable is created. The *setenv* command returns the new value of *variable_name*. To develop scripts that interact with the invoking shell, use *getenv* and *setenv*.

Environment variables are stored in the Tcl array variable *env*. The environment commands *getenv*, *setenv*, and *printenv* are convenience functions to interact with this array.

The *setenv* command sets the value of a variable only within the process of your current application. Child processes initiated from the application using the *exec* command after a usage of *setenv* inherit the new variable value. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the *exec* command, that value is not reflected in the current application.

Feedback

Examples

The following example changes the default printer.

```
shell> getenv PRINTER
laser1
shell> setenv PRINTER "laser3"
laser3
shell> getenv PRINTER
laser3
```

See Also

- getenv
- printenv
- printvar
- sh

---

sh

Executes a command in a child process.

Syntax

string *sh* [*args*]

```
string args
```

Arguments

args

Command and arguments that you want to execute in the child process.

Description

This is very similar to the *exec* command. However, file name expansion is performed on the arguments. Remember that quoting and grouping is in terms of Tcl. Arguments which contain spaces will need to be grouped with double quotes or curly braces. Tcl special characters which are being passed to system commands will need to be quoted and/or escaped for Tcl. See the examples below.

Examples

This example shows how you can remove files with a wildcard.

```
prompt> ls aaa*
aaa1      aaa2      aaa3
prompt> sh rm aaa*
```

```
prompt> ls aaa*
Error: aaa*: No such file or directory
       Use error_info for more info. (CMD-013)
```

This example shows how to grep some files for a regular expression which contains spaces and Tcl special characters:

```
prompt> exec cat test3.out
blah blah blah
blah blah blah c blah
input [1:0] A;
output [2:0] B;
prompt>
prompt> sh egrep -v {[ ]+c[ ]+} test3.out
blah blah blah
prompt>
prompt> sh egrep {t[ ]+\\[} test3.out
input [1:0] A;
output [2:0] B;
```

---

sh_list_key_bindings

Displays all the key bindings and edit mode of current shell session.

Syntax

string *sh_list_key_bindings*

```
[-nosplit]
```

Arguments

`-nosplit`

Prevents line splitting when column fields overflow.

Description

The *sh_list_key_bindings* command displays current key bindings and the edit mode. To change the edit mode, set the *sh_line_editing_mode* variable in either the .synopsys_pt.setup file or directly in the shell.

The text Ctrl+K is read as 'Control+K' and describes the character produced when you hold the Ctrl key while pressing the K key.

The text META+K is read as 'Meta+K' and describes the character produced when you hold down the Meta key while pressing the K key. The Meta key is labeled Alt on many keyboards. On keyboards with two keys labeled Alt (usually to either side of the space bar), the Alt on the left side is generally set to work as a Meta key. The Alt key on the right can also be configured to work as a Meta key or can be configured as some other modifier, such as a Compose key for typing accented characters.

If you do not have a Meta or Alt key, or another key working as a Meta key, the identical keystroke can be generated by pressing Esc first, and then typing K. Either process is known as metafying the K key.

Alternative key bindings work only in vi alternate(command) mode.

See Also

- sh_enable_line_editing
- sh_line_editing_mode

---

show_rule_violation

Displays the violation details for the given rule on the terminal.

Syntax

string *show_rule_violation*

```
        rule
```

Data Types

```
rule              string
```

Arguments

`-rule` *rule*

Specifies the name of an existing rule.

Description

Displays the violation details for each violation of a particular rule along with debugging help. To be used after executing the check_constraints command.

Examples

The following example displays all the violations for rule *EXD_0009*.

```
pt_shell> show_rule_violation -rule EXD_0009
```

See Also

- check_constraints

Feedback

sim_analyze_clock_network

Analyzes a clock network with a SPICE simulator.

Syntax

int *sim_analyze_clock_network*

```
-from from_pins
[-from_rise_slew rise_slew]
[-from_fall_slew fall_slew]
[-output model_file_name]
[-to to_pins]
[-use_probe]
[-min]
[-max]
```

Data Types

from_pins	collection
rise_slew	float
fall_slew	float
model_file_name	string
to_pins	collection

Arguments

`-from` *from_pins*

Specifies a collection of one root pin or port of the clock network. clock network is extracted by tracing from the root until sequential loads are reached. Clock gating cells are included in the clock network. It is essential for you to choose the appropriate root for clock network to satisfy their needs.

`-from_rise_slew` *rise_slew*

Specifies the rise slew at the clock root node in main library units. If specified, it is used for creating input stimulus. Otherwise, slew from PrimeTime is used.

`-from_fall_slew` *fall_slew*

Specifies the fall slew at the clock root node in main library units. If specified, it is used for creating input stimulus. Otherwise, slew from PrimeTime is used.

`-output` *model_file_name*

Specifies the file name in which the results of the simulations are written. It is written in Synopsys Design Constraint format using the *set_annotated_delay*, *set_annotated_transition* and *set_disable_timing* commands. This file is the clock mesh model that can be used for multiple runs on the same design of PrimeTime if clock network has not changed between runs.

`-to` *`to_pins`*

Specifies the collection of pins, when reached, stops the clock network tracing. If one of the load pins of the net is included, other load pins of the net are also automatically included to stop clock network tracing. This option helps control the simulation and back annotation of the clock network logic that is being analyzed.

`-use_probe`

Adds probe points to the SPICE simulation for all pins in the clock network.

`-min`

Runs and annotates only the min analysis.

`-max`

Runs and annotates only the max analysis.

Description

This command extracts the clock network from the specified clock root and invokes the simulator specified by the *sim_setup_simulator* command. The purpose of the command is to create a SPICE deck, link to simulation environment, and back-annotate the results into PrimeTime. The SPICE deck generated is sensitized using the same rules as the *write_spice_deck* command. The input stimulus at the clock root is created based on the clock slew reaching at the root node. If you specify the rise and fall slews, they are used to create the input stimulus. Otherwise, the default rise and fall slews from PrimeTime are used. Only a single clock network can be traced with the specified *from_pins* and *to_pins*.

This command triggers an implicit *update_timing* command to extract timing and sensitization information needed for generating SPICE deck of the clock network.

This command errors out when any of these features are enabled: crosstalk analysis, power analysis, variation analysis, multicore distributed analysis, HyperScale analysis.

The command also errors out when variable timing_reduce_multi_drive_net_arcs is enabled.

Unless the -min/-max options are used, sim_analyze_clock_network will, be default, run only the max analysis and annotate for both min and max.

To use the *sim_analyze_clock_network* command, you must have a PrimeTime SI license.

Examples

The following example analyzes the clock network starting from CLK port and creates a corresponding clock model in clock_model.tcl file.

```
pt_shell> sim_analyze_clock_network -from [get_ports CLK] \\
          -output clock_model.tcl
```

Feedback

See Also

- sim_setup_simulator
- sim_setup_library
- update_timing
- write_spice_deck

---

sim_analyze_path

Analyzes timing paths by performing SPICE simulation.

Syntax

status *sim_analyze_path*

```
[-sim_mode sim_mode]
[-num_sigma sigma_value]
[-sample_size value]
[-transition_time]
[-variation]
[-crosstalk_delta]
[-derate]
[-significant_digits digits]
[-verbose]
path_objects
```

Data Types

sim_mode	string
sigma_value	float
value	integer
digits	integer
path_objects	collection

Arguments

`-sim_mode` *sim_mode*

Specifies the parts of the timing path to be simulated and analyzed. It can be set to one of the following strings: *launch* (launch segment only), *capture* (capture segment only), or *both*. The default is *both*.

`-num_sigma` *sigma_value*

Specifies the sigma value that is the focus of analysis, from 2.0 to 6.0. If you do not use this option, the *timing_pocvm_report_sigma* variable setting is used, which is 3.0 by default. The *-num_sigma* option works only with the *-variation* option.

`-sample_size` *value*

Specifies the number of Monte Carlo samples used for POCV simulation. If you do not specify the sample size, the command automatically chooses a suitable number based on the *-num_sigma* option setting, if any, or the default sigma value otherwise.

`-transition_time`

Reports the simulated transition times as well as delays. By default, only delays are reported.

`-variation`

Reports POCV simulated corner values as well as nominal values. POCV analysis must be enabled to use this option. By default, only nominal values are reported.

`-crosstalk_delta`

Reports the PrimeTime annotated delta delay values at cell input pins in a column labeled "Delta". By default, delta delays are not reported.

`-derate`

Reports derating factors in a column labeled "Derate". By default, derating values are not reported.

`-significant_digits` *digits*

Specifies the number of digits after the decimal point displayed for time values in the generated report. The default value is 4.

`-verbose`

Reports more details, such as the location of temporary files.

path_objects

Specifies a collection of timing path objects to simulate; -pba_mode path is required. *-path_type full_clock_expanded* is recommended.

Description

The *sim_analyze_path* command performs HSPICE simulation on selected paths. This command is a PrimeShield feature, which requires a PrimeShield license. You must enable PrimeShield analysis (set *ps_enable_analysis* true) before you run simlink commands.

The *sim_analyze_path* command supports two variation simulation modes: full Monte Carlo and fast Monte Carlo. The default is full MC mode, which can require a very long

runtime. To enable the fast MC mode, set the *simlink_enable_fast_monte_carlo* variable to true:

```
pt_shell> set_app_var ps_enable_analysis true
Information: Checked out license 'PrimeShield' (PT-019)
true
pt_shell> set_app_var simlink_enable_fast_monte_carlo true
true
```

For cells that have a higher threshold voltage, the delay variation is larger and more MC samples are needed to make the simulation results converge to a real distribution at a high sigma corner. Fast MC speeds up simulation significantly, while getting accuracy similar to full MC.

By default, *sim_analyze_path* runs HSPICE simulation from the clock reconvergence pessimism (CRP) common point when the *-path_type full_clock_expanded* option is used in the *get_timing_paths* command. It is recommended to get timing paths with these options to make sure simulation is from the CRP point and the clock path is recalculated:

```
pt_shell> set_app_var pba_recalculate_full_path true
 ...
pt_shell> set mypaths \\
 [get_timing_paths -pba_mode path -path_type full_clock_expanded]
 ...
```

The *sim_analyze_path* command reports launch, capture, constraint, and slack using measured results from SPICE simulations whenever possible. When SPICE simulations are not applicable or not available, for example, zero-stage path without driving cell, crosstalk delta, constraint delay, derating factors, PrimeTime computed values are being used.

Before using this command, run the *sim_validate_setup* command to validate the SPICE simulation setup and environment. Incorrect setup often causes correlation problems.

Examples

The following example shows report with *sim_analyze_path* for nominal reporting.

```
pt_shell> sim_analyze_path -transition_time -crosstalk_delta -derate
 $mypaths
Analyzing path number    0 :
  Startpoint: f1 (rising edge-triggered flip-flop clocked by clk1)
  Endpoint: f2 (rising edge-triggered flip-flop clocked by CLK3)
  CRP point: i55/ZN
Information: Generate spice deck from CRP point i55/ZN to end point.
  Launch simulation starts from i55/ZN
  Capture simulation starts from i55/ZN
Launch path segment:
  Pin                Sense   Transition     Derate      Delta        Incr
       Path
```

```
------------------------------------------------------------------------
-------------
  clock clk1 (rise edge)                                          0.0000
      0.0000
  i55/ZN                  r       0.0425                          0.0000
      0.0000
  f1/CP                   r       0.0425     1.1000     0.0000    0.0000
      0.0000
  f1/Q                    f       0.7983     1.0000     0.0000    0.5301
      0.5301
  i7/I                    f       0.7972     1.1000     0.6071    0.7094
      1.2395
  i7/ZN                   r       0.5695     1.0000     0.0000    0.6786
      1.9181
  i8/I                    r       0.5801     1.1000     0.1090    0.1573
      2.0754
  i8/ZN                   f       0.0717     1.0000     0.0000    0.1224
      2.1978
  f2/D                    f       0.0717     1.1000     0.0000    0.0000
      2.1978
------------------------------------------------------------------------
-------------
  Arrival
      2.1978

Capture path segment:
  Pin                Sense   Transition     Derate      Delta        Incr
       Path
------------------------------------------------------------------------
-------------
  clock CLK3 (rise edge)                                          2.0000
      2.0000
  i55/ZN                  f       0.0422                          0.0000
      2.0000
  i56/I                   f       0.0422     0.9000     0.0000    0.0000
      2.0000
  i56/ZN                  r       0.0117     1.0000     0.0000    0.0226
      2.0226
  f2/CP                   r       0.0117     0.9000     0.0000    0.0000
      2.0226
------------------------------------------------------------------------
-------------
  Required
      2.0226
Constraint:
------------------------------------------------------------------------
-------------
  setup time                                  1.0000               -0.0154
      2.0073
------------------------------------------------------------------------
-------------
  Slack
     -0.1905
```

The following example shows report with *sim_analyze_path* using fast MC for variation.

```
pt_shell> sim_analyze_path -variation -crosstalk_delta $mypaths
Information : Fast MC runtime: 171 seconds.
Information : Fast MC capture runtime: 24 seconds.
  Analyzing path number    0 :
  Startpoint: f1 (rising edge-triggered flip-flop clocked by clk1)
  Endpoint: f2 (rising edge-triggered flip-flop clocked by CLK3)
  CRP point: i55/ZN
Information: Generate spice deck from CRP point i55/ZN to end point.
  Launch simulation starts from i55/ZN
  Capture simulation starts from i55/ZN
Launch path segment:
Confidence interval: (1.5692, 1.5888); accuracy target: 0.62%.
                                          ------- Incr --------   --------
 Path -------
  Pin                Sense      Delta     Nominal      Corner     Nominal
      Corner
-------------------------------------------------------------------------
-------------
  clock clk1 (rise edge)                    0.0000      0.0000      0.0000
      0.0000
  i55/ZN                 r                  0.0000      0.0000      0.0000
      0.0000
  f1/CP                  r     0.0000       0.0000      0.0000      0.0000
      0.0000
  f1/Q                   f     0.0000       0.5303      0.6603      0.5303
      0.6603
  i7/I                   f     0.6071       0.7106      0.7106      1.2410
      1.3709
  i7/ZN                  r     0.0000       0.6771      0.7946      1.9180
      2.0932
  i8/I                   r     0.1090       0.1573      0.1573      2.0753
      2.2505
  i8/ZN                  f     0.0000       0.1225      0.1514      2.1978
      2.3754
  f2/D                   f     0.0000       0.0000      0.0000      2.1978
      2.3754
-------------------------------------------------------------------------
-------------
  Arrival                                                            2.1978
      2.3754

Capture path segment:
Confidence interval: (0.0198, 0.0201); accuracy target: 0.78%.
                                          ------- Incr --------   --------
 Path -------
  Pin                 Sense     Delta     Nominal      Corner     Nominal
      Corner
-------------------------------------------------------------------------
-------------
  clock CLK3 (rise edge)                    2.0000      2.0000      2.0000
      2.0000
```

```
  i55/ZN                                    0.0000      0.0000      2.0000
      2.0000
  i56/I                          -0.0000    0.0000      0.0000      2.0000
      2.0000
  i56/ZN                         -0.0000    0.0227      0.0200      2.0227
      2.0200
  f2/CP                          -0.0000    0.0000      0.0000      2.0227
      2.0200
-------------------------------------------------------------------------------
-------------
  Required                                                           2.0227
      2.0200
Constraint:
-------------------------------------------------------------------------------
-------------
  setup time                                -0.0154     -0.0154      2.0074
      2.0046
-------------------------------------------------------------------------------
-------------
  Slack                                                             -0.1905
     -0.3681

Constraint:
-------------------------------------------------------------------------------
-------------
  setup time                                -0.0154     -0.0154      2.0100
      2.0068
-------------------------------------------------------------------------------
-------------
  Slack                                                             -0.1868
     -0.3631
```

See Also

- sim_setup_library
- sim_setup_simulator
- simlink_enable_fast_monte_carlo
- ps_enable_analysis

---

sim_corruption_control

Provides the ability to disable the corruptions of a specific set of design elements or types of design elements.

Syntax

string *sim_corruption_control*

```
[-type real | integer | seq | comb | port | process]
[-elements list]
[-exclude_elements list]
[-models model_list]
[-domain domain_name]
[-transitive TRUE | FALSE]
```

Data Types

list	list
model_list	list
domain_name	string
transitive	boolean

Arguments

`-type real | integer | seq | comb | port | process`

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-elements list`

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-exclude_elements list`

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-models model_list`

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-domain string`

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-transitive TRUE | FALSE`

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

Description

This command exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

Feedback

Examples

The following command selects all real nets that start with L but excludes those that start with L1.

```
prompt> sim_corruption_control -type real
-elements [find_objects . -pattern "L*" -object_type net]
-exclude_elements [find_objects . -pattern "L1*" -object_type net]
1
```

See Also

- sim_replay_control

---

sim_enable_si_correlation

Enables signal integrity correlation mode for specific nets.

Syntax

status *sim_enable_si_correlation*

net_list

Data Types

net_list list

Arguments

net_list

Specifies a list of nets for signal integrity SPICE correlation of coupled stage delay or noise. The *net_list* accepts a maximum of 1000 nets.

Description

This command specifies nets for signal integrity SPICE correlation.

To use this command, you must

- Enable crosstalk analysis by setting *si_enable_analysis* to *true*.
- Have PrimeTime SI license
- Follow the recommended settings for SPICE correlation

Feedback

Examples

The following example enables signal integrity correlation on a net and validates coupled stage accuracy on the corresponding net arc.

```
pt_shell> sim_enable_si_correlation [get_nets -of_objects I1/Z]
pt_shell> update_timing -full
pt_shell> sim_validate_stage [get_timing_arc -from I1/Z -to I2/A]
```

The following example enables signal integrity correlation on a net and validates noise bump height and area accuracy on the corresponding net arc.

```
pt_shell> sim_enable_si_correlation [get_nets -of_objects I1/Z]
pt_shell> update_timing -full
pt_shell> update_noise -full
pt_shell> sim_validate_noise -area [get_timing_arc -from I1/Z -to I2/A]
```

See Also

- sim_setup_library
- sim_setup_simulator
- sim_validate_noise
- sim_validate_stage
- update_noise
- update_timing
- write_spice_deck
- si_enable_analysis

---

sim_replay_control

Specify initial blocks to be replayed when a domain powers up.

Syntax

string *sim_replay_control*

```
[-elements list]
[-exclude_elements list]
[-models model_list]
[-domain domain_name]
[-controlling_domain domain_name]
[-transitive TRUE | FALSE]
```

Feedback

Data Types

list	list
model_list	list
domain_name	string
controlling_domain	string
transitive	boolean

Arguments

`-elements` *list*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-exclude_elements` *list*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-models` *model_list*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-domain` *string*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-controlling_domain` *string*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

`-transitive` *TRUE* | *FALSE*

This option exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

Description

This command exists for compatibility with the Design Compiler and IC Compiler tools. PrimeTime reads and ignore this option.

Examples

The following command selects all initial blocks whose label starts with L but excludes those blocks whose label starts with L1.

```
prompt> sim_replay_control
-elements [find_objects . -pattern "L*" -object_type process]
-exclude_elements [find_objects . -pattern "L1*" -object_type process]
1
```

Feedback

See Also

- sim_corruption_control

---

sim_setup_distributed

Specifies the host options to use for launching HSPICE simulations of multiple paths in parallel.

Syntax

status *sim_setup_distributed*

```
[-mode lsf | csh | grd | nc]
[-sim_per_job number]
[-farm_max_num_pending number]
[-max_job_requeue number]
[-farm_timeout number]
[-submit_command command]
[-pre_run_command command]
[-host_option_name host_option_name]
[-reset]
```

Data Types

number	integer
command	string
host_option_name	string

Arguments

`-mode lsf | csh | grd | nc`

Specifies the protocol used to launch the jobs.

`-sim_per_job number`

Specifies the number of simulations per submitted farm job.

`-farm_max_num_pending number`

Specifies the maximum number of pending jobs to allow in the farm queue.

`-max_job_requeue number`

Specifies the maximum number of times a failed job is requeued.

`-farm_timeout number`

Specifies the timeout limit for the farm run, in seconds.

`-submit_command command`

Specifies the submit command and arguments used to submit jobs to the farm.

`[-pre_run_command command]`

Specifies a pre-run command to run before simulation.

`[-host_option_name host_option_name]`

References a host configuration definited using the *set_host_options -name* command instead.

`[-reset]`

Resets all settings in the command to default values.

See the "Using CDPL Jobs" section for details.

Description

This command is used to configure how simulation jobs are launched when performing parallel HSPICE simulation of multiple paths in a timing path collection.

The *sim_setup_distributed* command must be specified before any SimLink commands that run the simulation and generate reports (such as the *sim_validate_path*, *sim_validate_stage*, and *sim_validate_noise* commands). Note that the *sim_setup_distributed* command does not support the *-from* or *-to* options of the *sim_validate_path* command.

Additional options of the *sim_setup_distributed* command allow the tool to monitor farm job status, such as the number of jobs still in the queue (not running), the time such jobs are allowed to remain in the queue, and how many times a failed job is resubmitted. The *-pre_run_command* option allows you to specify commands to be run prior to submitting the jobs.

After a SimLink command completes, it reports a status summary of the submitted jobs.

Using CDPL Jobs

The *sim_setup_distributed* command also allows you to use processing resources that are configured by CDPL (Common Distributed Processing Library), which is a standardized way of configuring job resources using the *set_host_options* command.

To do this, define compute resources using the *set_host_options -name* command then reference those named resources with the *-host_option_name -host_option_name* command. The *set_host_options* command must also use the *-target Hspice* option to indicate that the jobs will be used for simulation.

Here is an example of how to set up CDPL on compute farm machines:

```
set simlink_enable_fast_monte_carlo true
set simulator "/path/to/hspice/bin/hspice"

# start CDPL processes
set_host_options -target Hspice -name my_compute_farm_jobs \\
```

```
  -num_processes 10 -max_cores 8 \\
  -submit_command "qsub -P typ -V -cwd -l mem_free=10G"
start_hosts -min_hosts 1

# use CDPL processes for simulation
sim_setup_distributed -host_option_name my_compute_farm_jobs

sim_setup_simulator -simulator $simulator -simulator_type hspice \\
  -work_dir sim_dir -preserve all
sim_setup_library -lib [get_libs *] -sub spice_netlist -header
 spice_header
sim_analyze_path -variation -sample_size 10000 $paths
stop_hosts
```

Here is an example of how to set up CDPL on specific machines:

```
set simlink_enable_fast_monte_carlo true
set simulator "/path/to/hspice/bin/hspice"

# start CDPL processes
set_host_options -target Hspice -name my_direct_jobs \\
  -num_processes 10 -max_cores 8 {host1 host2 host3}
start_hosts -min_hosts 1

# use CDPL processes for simulation
sim_setup_distributed -host_option_name my_direct_jobs

sim_setup_simulator -simulator $simulator -simulator_type hspice \\
  -work_dir sim_dir -preserve all
sim_setup_library -lib [get_lib *] -sub spice_netlist -header
 spice_header
sim_analyze_path -variation -sample_size 10000 $paths
stop_hosts
```

Examples

The following example configures 10 simulations to be run in each farm job.

```
pt_shell> sim_setup_distributed -mode grd -sim_per_job 10 \\
            -submit_command {qsub -V -cwd -P typ -l mem_free=10G}
```

Consider a case where you want to run *sim_validate_path -variation* for a path collection of 100 paths. Because Monte Carlo simulation is slow, you want each job to run only one Monte Carlo simulation. In this case, use the *-sim_per_job 1* option, so that the tool launches individual Vsimulations for all paths in parallel:

```
pt_shell> sim_setup_distributed -mode grd -sim_per_job 1 \\
            -submit_command {qsub -V -cwd -P typ -l mem_free=10G}
```

To run *sim_validate_path -variation* simulations at high sigma values, you can use the HSPICE distributed processing *-dp* option to speed up fast Monte Carlo simulation for each path. It is recommended to use *-dp 8* to run fast Monte Carlo at high sigma values.

Feedback

```
pt_shell> set simulator "/2018.09-SP2-2/bin/hspice -dp 8 \\
            -dpconfig /abc/dpconfig -i"
pt_shell> sim_setup_distributed -mode grd -sim_per_job 1 \\
            -submit_command {qsub -V -cwd -P typ -l mem_free=10G}
```

Here is an example dpconfig file:

```
#flag|hostname|slots|tmpdir|protocal|command
    1|        | -1  | /tmp | SGE    |qsub -P typ -cwd -V -l mem_free=10G
#   1|        | -1   /tmp  | LSF    |bsub -q typ -R "rusage[mem=1000]"
```

See Also

- set_host_options
- sim_validate_path
- sim_validate_noise
- sim_validate_stage
- sim_analyze_path

---

sim_setup_library

Sets up files for a gate-level library used in SPICE circuit simulation.

Syntax

status *sim_setup_library*

```
-library library
-sub_circuit dir
-header file
[-file_name_pattern pattern]
```

Data Types

```
library     collection
dir         string
file        string
pattern     string
```

Arguments

`-library library`

Specifies the PrimeTime gate-level library to be used in SPICE simulation.

`-sub_circuit` *dir*

Specifies the directory that contains the subcircuit files (one file per cell) that are included in the simulation SPICE decks. Specifying a single file instead of a directory is allowed but might result in poor simulation performance.

`-header` *file*

Specifies a file to be included at the beginning of the SPICE file. This file must contain a model file, corner instantiation, and simulator options. The file must not contain any measure statement or other circuit-specific statements. Generally, the SPICE options must be the same in all header files for each library.

Temperature information must not be present in the header files; this is automatically added to the SPICE deck by PrimeTime.

By default, the power rail name is VDD (which uses the main library voltage) and the ground rail name is VSS (which uses 0 volts). To define rails other than VDD and VSS, specify the rail voltage definitions in the header files.

`-file_name_pattern` *pattern*

Restricts the files used for this library to those that follow the specified file name pattern. This option is helpful if the directory specified by the *-sub_circuit dir* option contains subcircuits for multiple libraries.

Description

This command performs setup for the gate-level library in preparation for SPICE simulation, such as mapping transistor models to gate-level models.

When the *simlink_enable_fast_monte_carlo* variable is set to *true*, the *sim_setup_library* command performs library-level learning for fast Monte Carlo analysis. This requires the *sim_setup_simulator* command to be run before the *sim_setup_library* command.

Examples

The following example associates a specified library with a SPICE subcircuit directory and header file.

```
pt_shell> sim_setup_library -library my_lib \\
          -sub_circuit /path/gem/hspice \\
          -header /path/gem/model_hspice \\
          -file_name_pattern {my_%s.spc}
```

In this example, the IV170BQ library cell uses the my_IV170BQ.spc file in the subcircuit directory to get its subcircuit.

The *-file_name_pattern* option helps in associating the library cell names to my_{lib_cell_name}.spc files in the subcircuit directory.

Feedback

See Also

- sim_analyze_clock_network
- sim_setup_simulator
- simlink_enable_fast_monte_carlo

---

sim_setup_simulator

Sets up the simulation environment.

Syntax

status *sim_setup_simulator*

```
[-simulator sim_exec_path]
[-simulator_type hspice | nanosim | hsim]
[-sim_options option_string]
[-work_dir directory]
[-preserve failed | all | none]
```

Data Types

```
sim_exec_path      string
option_string      string
directory          string
```

Arguments

`-simulator sim_exec_path`

Specifies the path to the simulation executable.

`-simulator_type hspice | nanosim | hsim | customsim`

Specifies one of the following simulators

- *hspice* (default) - HSPICE
- *nanosim* - NanoSim
- *hsim* - HSIM
- *customsim* - CustomSim

If you do not use this option, PrimeTime tries to derive the simulator type by using the *-simulator* option. If PrimeTime cannot derive the simulator type, the default simulator is HSPICE. CustomSim is not recommanded for simlink commands for correlation purpose.

Feedback

`-sim_options` *option_string*

Specifies the default simulator options that are appended to the SPICE decks.

`-work_dir` *directory*

Specifies the directory that stores temporary SPICE files. This option overrides any previous *-work_dir* options -- that is, there is only one for all libraries. The default directory is $pt_tmp_dir.

`-preserve failed | all | none`

Specifies whether temporary files, such as SPICE decks, are kept on the disk.

- *all* - Preserves all files.
- *none* - Deletes all files.
- *failed* (default) - Keeps the files only if simulation failed.

Description

This command sets up the simulation environment. It specifies the simulator location, the type of simulator, and the working directory.

Examples

The following example sets the simulator exec, simulator type, and the default options for the simulator.

```
pt_shell> sim_setup_simulator -simulator /global/apps/hspice/bin/hspice
 \\
          -simulator_type hspice \\
          -sim_options ".option ingold=2 statfl=1 acct=0 autostop brief
 runlvl=5"
```

The following example sets the work directory and marks all the temporary files to be preserved after simulation has finished.

```
pt_shell> sim_setup_simulator -work_dir [pwd] \\
          -preserve all
```

See Also

- sim_analyze_clock_network
- sim_setup_library
- sim_validate_noise
- sim_validate_path

Feedback

- sim_validate_setup
- sim_validate_stage

---

sim_setup_spice_deck

Specifies setup options for SPICE deck generation.

Syntax

status *sim_setup_spice_deck*

```
[-enable_clock_mesh]
[-use_pin_load]
```

Arguments

`-enable_clock_mesh`

Enables the mesh model generation flow for performing clock network analysis using the *sim_analyze_clock_network* command. Set this option to enable the clock model generation flow before reading the parasitics of the design. Otherwise, there is a possibility of a few large multidriven nets (mesh nets) being treated as ideal nets.

`-use_pin_load`

Replaces all the side-load cell instances by its input pin capacitance. The input pin capacitance used is nonlinear delay model (NLDM) pin capacitance of the lib cell specified in the library. CCS pin capacitances (c1, c2) are ignored and an equivalent pin cap is used in the SPICE deck.

This option is beneficial in some cases because it improves the SPICE simulation runtime and also enables the SPICE simulations for load cells that do not have corresponding SPICE models. However, using this option results in loss of accuracy of SPICE simulation because the physical load cells are replaced by its equivalent pin cap model. This feature is also honored by the SPICE deck that is generated by the *write_spice_deck* command.

Description

This command specifies setup options for writing out the SPICE deck.

Examples

The following example enables clock mesh model generation flow.

```
pt_shell> sim_setup_spice_deck -enable_clock_mesh
```

The following example replaces all the side-loads in the SPICE deck with the input pin capacitance.

Feedback

```
pt_shell> sim_setup_spice_deck -use_pin_load
```

See Also

- sim_analyze_clock_network
- write_spice_deck

---

sim_validate_noise

Validates PrimeTime noise analysis accuracy compared with SPICE simulation.

Syntax

int *sim_validate_noise*

```
[-analysis_type above_low | below_high]
[-area]
[-verbose]
[-significant_digits digits]
net_arc_objects
```

Data Types

```
net_arc_objects    collection
```

Arguments

`-analysis_type above_low | below_high`

Reports the specified type of correlation. By default, both types of correlation are reported.

`-area`

Shows the noise bump area comparison results. By default, only noise bump heights are compared.

`-verbose`

Reports additional information, such as the location of temporary files.

`-significant_digits digits`

Specifies the number of digits after the decimal point displayed for timing values in the generated report. Allowed values are 0-13; the default is determined by the *simlink_report_default_significant_digits* variable, which is 4 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

net_arc_objects

Specifies a collection of net *timing_arc* objects to validate.

Description

The *sim_validate_noise* command simulates the specified net noise effects, comparing the bump height (and area) between PrimeTime and simulation. The error percentage in the report are reported as a percentage of VDD.

Before using this command, run the *sim_validate_setup* command to validate the SPICE simulation setup and environment. Incorrect setup of the simulation environment often causes correlation problems.

Then, run the *sim_enable_si_correlation* command before the *update_timing* and *update_noise* commands on the nets of interest, and follow the recommended settings for SPICE correlation.

This command is for noise correlation only. For uncoupled delay correlation, use the *sim_validate_path* command. For coupled delay correlation, use the *sim_validate_stage* command.

To use the *sim_validate_noise* command, you must have a PrimeTime SI license.

Examples

The following example simulates noise at U2/A, based on the U1/Z to U2/A net arc. It compares both bump height and area results with SPICE for *above_low*.

```
pt_shell> sim_enable_si_correlation [get_nets -of_objects U1/Z]
pt_shell> update_timing -full
pt_shell> update_noise -full
pt_shell> set net_arc [get_timing_arc -from U1/Z -to U2/A]
pt_shell> sim_validate_noise $net_arc -area -analysis_type above_low
```

See Also

- sim_enable_si_correlation
- sim_setup_library
- sim_setup_simulator
- sim_validate_path
- sim_validate_setup
- sim_validate_stage
- simlink_report_default_significant_digits

Feedback

sim_validate_path

Validates uncoupled PrimeTime delay and transition accuracy for a timing path using SPICE simulation.

Syntax

status *sim_validate_path*

```
[-from start_pin_object]
[-to end_pin_object]
[-transition_time]
[-verbose]
[-variation]
[-sample_size number_of_mc_samples]
[-significant_digits digits]
[-sep_init]
path_objects
```

Data Types

start_pin_object	collection
end_pin_object	collection
number_of_mc_samples	integer
path_objects	collection

Arguments

`-from` *start_pin_object*

Specifies the start pin object for simulation.

`-to` *end_pin_object*

Specifies the end pin object for simulation.

`-transition_time`

Shows the transition time comparison results. By default, only delays are compared.

`-verbose`

Reports additional information, such as the location of temporary files.

`-variation`

Reports POCV correlation for nominal and corner values of path arrival. POCV analysis must be enabled to use this feature. If the *-from* and *-to* options are specified, the variation correlation is reported for the path segment between *start_pin_object* and *end_pin_object* on *path_objects*.

`-sample_size` *`number_of_mc_samples`*

Specifies the number of Monte Carlo samples for POCV correlation. The default is 5000 samples. This option has an effect only when used with the *-variation* option.

`-significant_digits` *`digits`*

Specifies the number of digits after the decimal point displayed for timing values in the generated report. Allowed values are 0-13; the default is determined by the *simlink_report_default_significant_digits* variable, which is 4 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

`-sep_init`

Separates the initialization SPICE simulation from the measurement simulation. This can reduce the chance of measurement failure when complicated sequential cells are involved.

`path_objects`

Specifies a collection of timing path objects to validate.

Description

The *sim_validate_path* command simulates the specified path or path segment and compares the delays (and optionally transition times) between PrimeTime analysis and SPICE simulation.

Before using this command, run the *sim_validate_setup* command to validate the SPICE simulation setup and environment. Incorrect setup often causes correlation problems. Use path-based PrimeTime analysis for correlation and follow the recommended settings for SPICE correlation.

This command works for uncoupled correlation only. In designs with cross-coupling, a *SIM-004* error is issued. For paths with cross-coupling, use the *sim_validate_stage* command.

For POCV validation with *sim_validate_path -variation*, it is highly recommended to enable fast Monte Carlo SPICE simulation by setting the *simlink_enable_fast_monte_carlo* variable to *true*, which significantly reduces POCV validation runtime.

Examples

The following example performs path-based analysis, sets a variable to a collection containing the worst path, and compares the delay and transition analysis results with SPICE simulation of the path.

Feedback

```
pt_shell> set pba_path [get_timing_paths -pba_mode path]
pt_shell> sim_validate_path $pba_path -transition
```

The following example analyzes a path segment of interest (from U1/A to U2/Z) and compares the PrimeTime delay results with SPICE simulation results.

```
pt_shell> set pba_path [get_timing_path -through U1/A \\
            -through U2/Z -pba_mode path]
pt_shell> sim_validate_path $pba_path -from U1/A -to U2/Z
```

The following example performs path-based analysis, sets a variable to a collection containing the worst path, and compares path arrival nominal and corner value with fast Monte Carlo simulated results.

```
pt_shell> set_app_var simlink_enable_fast_monte_carlo true
pt_shell> sim_setup_simulator -simulator /global/apps/hspice/bin/hspice
pt_shell> sim_setup_library -lib [get_libs *] -sub_circuit sub_path
 -header header_path
pt_shell> set pba_path [get_timing_paths -pba_mode path -path_type
 full_clock_expanded]
pt_shell> sim_validate_path $pba_path -variation
```

See Also

- sim_setup_library
- sim_setup_simulator
- sim_validate_setup
- sim_validate_stage
- simlink_enable_fast_monte_carlo
- simlink_report_default_significant_digits

---

sim_validate_setup

Validates SPICE simulation setup and environment.

Syntax

status *sim_validate_setup*

```
-lib_cell library_cell
[-from from_pin]
[-to to_pin]
[-capacitance cap_value]
[-transition_time tran_value]
[-significant_digits digits]
[-verbose]
```

Data Types

library_cell	collection
from_pin	string
to_pin	string
cap_value	float
tran_value	float

Arguments

`-lib_cell` *library_cell*

Specifies the library cell object to verify

`-from` *from_pin*

Specifies the from pin of the library arc. The tool uses the library arc specified by the *-from* and *-to* options for setup verification.

`-to` *to_pin*

Specifies the to pin of the library arc. The tool uses the library arc specified by the *-from* and *-to* options for setup verification.

`-capacitance` *cap_value*

Specifies the output load to be driven by the library cell. The units are in PrimeTime main library units.

`-transition_time` *tran_value*

Specifies the input transition time (slew) for the arc. A ramp or CCS standard pre-driver waveform (as per the library driver waveform) with this slew is generated at the corresponding input pin of this library cell. Both rise and fall will be simulated. The units are in PrimeTime main library units.

`-significant_digits` *digits*

Specifies the number of digits after the decimal point displayed for timing values in the generated report. Allowed values are 0-13; the default is determined by the *simlink_report_default_significant_digits* variable, which is 4 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

`-verbose`

Shows more details such as location of temporary files.

Description

Incorrect user setup of the simulation environment often causes problems. The goal of this command is to help you ensure that the PrimeTime libraries and SPICE environment are matching. This command invokes the simulator specified by the

Feedback

sim_setup_simulator command on the given library cell arcs. The purpose is to verify that the basic characterization of a library matches the SPICE setup specified by the *sim_setup_** commands. To make the interface as simple as possible, only combinational cells are supported. If any error is detected in the SPICE run, FALSE is returned to the Tcl interpreter. Otherwise, the command displays the SPICE and PrimeTime computed delay and slew values, and returns TRUE.

This command checks for the presence of SPICE executable, SPICE netlist, SPICE model files, SPICE options, I/O rail settings, and correct units. It also checks the model license and compatibility of the SPICE version with the model.

This command does not replace Library checker or validation. The usage model is the other way around: Library has to be fully validated for accuracy first. This command assumes that the Liberty library is 100% correct and accurate, and therefore any PrimeTime and SPICE mismatch is due to incorrect simulation setup.

You can use this command only when there are no designs loaded in PrimeTime. Run this command after reading the libraries. The SPICE validation check is required one time for each library generally, for the first time the SPICE environment is setup in the design flow with the specific library.

To use the *sim_validate_setup* command, you must have a PrimeTime SI license.

Examples

The example validates the simulation setup using a library arc from A to Y pins of IV170BQ lib cell with an output load of 48 library units and transition time of 0.2 units of library units.

```
pt_shell> sim_validate_setup -from A -to Y \\
          -lib_cell [get_lib_cells $lib_name/IV170BQ] \\
          -capacitance 48 -transition_time 0.2
```

See Also

- sim_setup_library
- sim_setup_simulator
- simlink_report_default_significant_digits

---

sim_validate_stage

Validates coupled PrimeTime accuracy compared with SPICE simulation.

Syntax

```
status sim_validate_stage
   [-verbose]
```

```
[-analysis_type min_rise | max_rise | min_fall | max_fall]
[-measure_type stage | lookahead]
[-significant_digits digits]
net_arc_objects
```

Data Types

net_arc_objects collection

Arguments

-verbose

Reports more details, such as the location of temporary files.

-analysis_type min_rise | max_rise | min_fall | max_fall

Specifies the analysis type of correlation, where "rise" or "fall" indicates a rising or falling transition on net arc load pin. If you do not specify this option, all four types of correlation are reported.

-measure_type lookahead | stage

Specifies the measure type of correlation:

- *lookahead* (the default) - Specifies one-and-half stage correlation.
- *stage* - Specifies stage correlation. Use this option for investigating outliers. When using this option, set the *si_ccs_aggressor_alignment_mode* variable to *stage* for correlation results.

-significant_digits *digits*

Specifies the number of digits after the decimal point displayed for timing values in the generated report. Allowed values are 0-13; the default is determined by the *simlink_report_default_significant_digits* variable, which is 4 by default. Use this option if you want to override the default for the current report. This option controls only the number of digits displayed, not the precision used internally for analysis.

net_arc_objects

Specifies a collection of net timing_arc objects to validate.

Description

The *sim_validate_stage* command simulates the specified stage and half and compares the delay between PrimeTime and simulation. For a net arc without receiver timing arc, stage correlation is reported.

Before using this command, run the *sim_validate_setup* command to validate the SPICE simulation setup and environment. Incorrect setup of the simulation environment often causes correlation problems. Run the *sim_enable_si_correlation* command before the

update_timing command on nets of interest, and follow recommended settings for SPICE correlation.

This command works for coupled correlation purpose only. For uncoupled correlation, use the *sim_validate_path* command.

To use the *sim_validate_stage* command, you must have a PrimeTime SI license.

Examples

The following example simulates one and half stage including U1/A to U1/Z cell arc, U1/Z to U2/A net arc, and U2/A to U2/Z cell arc. It compares delay results with SPICE for max_rise.

```
pt_shell> sim_enable_si_correlation [get_nets -of_objects U1/Z]
pt_shell> update_timing -full
pt_shell> set net_arc [get_timing_arc -from U1/Z -to U2/A]
pt_shell> sim_validate_stage $net_arc -analysis_type max_rise
```

See Also

- sim_enable_si_correlation
- sim_setup_library
- sim_setup_simulator
- sim_validate_path
- sim_validate_setup
- si_ccs_aggressor_alignment_mode
- simlink_report_default_significant_digits

---

size_cell

Relinks one or more leaf cell instances to a new library cell that has the same logical function but different properties such as size and drive strength.

Syntax

status *size_cell*

```
[-current_library]
[-libraries lib_spec]
[-all_mim_instances]
cell_list
lib_cell
```

Data Types

lib_spec	list
cell_list	list
lib_cell	string

Arguments

`-current_library`

Restricts selection of the new library cell to the same library as that of the cell being replaced. You can use the this option only if the *lib_cell* argument specifies a base cell name, without a library name.

`-libraries` *lib_spec*

Restricts selection of the new library cell to the specified list of libraries. The libraries are searched in the order in which they are listed. You can specify either a list of library names or a collection created by the *get_libs* command. You can use the this option only if the *lib_cell* argument specifies a base cell name, without a library name.

If you specify a base cell name and do not use the *-current_library* or *-libraries* option, the command looks for the new library cell in the current library first, then in the libraries specified by the *link_path_per_instance* variable, and finally in the libraries specified by the *link_path* variable. It uses the first matching library cell found.

`-all_mim_instances`

Relinking of one or more leaf cell instances to a new library cell in one MIM instance is replicated in all instances of a MIM set.

cell_list

Specifies one or more leaf cell instances to be relinked: a cell instance name, a list of cell instance names, or a cell collection gathered with the *get_cells* command. Each cell must be in the scope at or below the current instance.

lib_cell

Specifies the single library cell to which the specified cell instances are linked. It can be a simple base cell name such as "AND2a04", a full library/cell name such as "CBmax120/AND2a04", or a single library cell object collected by the *get_lib_cells* command. If you specify a simple base name, you can restrict the choice of libraries with the *-current_library* or *-libraries* option.

If you invoke *pt_shell* with the *-multi_scenario* option, you must specify only the base name of the library cell.

Feedback

Description

This command changes the drive strength or other properties of a leaf cell by relinking it to a new library cell that has the required properties and the same logical functionality.

You can get a list of library cells that are compatible with a given cell instance by using the *get_alternative_lib_cells* command. To report the timing effects of these compatible library cells, use the *report_alternative_lib_cells* command.

The *size_cell* command is only for leaf cells. To swap a hierarchical block for a different design, use the *swap_cell* command.

The *size_cell* command typically triggers an incremental timing update. However, a full timing update is required if the resized cell is in the clock tree or connected to the clock tree.

The *size_cell* command with *-all_mim_instances"* option applies changes to all the instances of the MIM set.

Suppose a CPU module is instantiated four times in the TOP module as CPU1, CPU2, CPU3 and CPU4.

The following example shows *size_cell* command being used in one instance of a MIM set i.e CPU1 with *-all_mim_instances"* option. The changes are replicated to all the other MIM instances of the set i.e CPU2, CPU3, CPU4.

```
pt_shell> size_cell CPU1/n1 mylib/NR4P -all_mim_instances"
Sized 'CPU1/n1' with 'mylib/NR4P'
Sized 'CPU2/n1' with 'mylib/NR4P'
Sized 'CPU3/n1' with 'mylib/NR4P'
Sized 'CPU4/n1' with 'mylib/NR4P'
1
```

Equivalent Library Cells

The *size_cell* command works only with library cells that are functionally equivalent to the cell being sized, in the following ways:

- Number of input pins
- Number of output pins
- Logical functionality
- Number, type, and sense of enabled cell timing arcs

By default, the I/O pins do *not* need to have the same names or ordering. Furthermore, the *size_cell* command recognizes a 2-input NAND gate and a 2-input OR gate, each cell with one negated input, as functionally equivalent (by DeMorgan's theorem), even if the replacement cell requires swapping the two input connections by name.

To enforce identical pin naming, set the *eco_strict_pin_name_equivalence* variable to true. In that case, the *size_cell* command considers matching pin names and their respective functions as an additional requirement for equivalence.

To disable the requirement that timing arcs must match between the target library cell and the original library cell, set the *eco_strict_lib_arc_equivalence* variable to *false*.

To determine functional equivalence, the *size_cell* command considers three lib_cell attributes created by the Library Compiler tool: *function_id*, *mog_func_id*, and *user_function_class*. If the attributes are present, they must match between the cell being resized and the replacement cell. The *mog_func_id* attribute is present only for a cell with multiple output gates. The *user_function_class* attribute is present only for functionally complex cells.

The *get_alternative_lib_cells* and *report_alternative_lib_cells* commands use the same criteria to determine functional equivalence.

Writing the Changes

The changes caused by netlist editing commands such as *size_cell* are recorded in a change list, which is annotated on the design. To view or export the change list, use the *write_changes* command.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design.

Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary, for example, when the design is relinked.

The tool automatically uniquifies the blocks as needed, including the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 mylib/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
```

```
Sized 'i1/low/n1' with 'mylib/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows how to query the *is_edited* attribute on i1/low.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Examples

In the following example, an attempt to size a cell fails because the target is not functionally equivalent. After finding a compatible library cell, the second attempt succeeds.

```
pt_shell> size_cell o_reg1 mylib/NR4P
Error: Could not size 'o_reg1' ('mylib/FD2') with 'mylib/NR4P':
       Cells not functionally equivalent. (NED-005)
Error: No changes made. (NED-040)
0

pt_shell> get_alternative_lib_cells o_reg1
{"mylib/FD2P"}

pt_shell> size_cell o_reg1 mylib/FD2P
Information: Sized 'o_reg1' with 'mylib/FD2P'. (NED-045)
1
```

The following example sizes two cell instances to alternative library cells using only the cell's current library and the library cell base name.

```
pt_shell> get_lib_cells -of_objects n1
{"mylib/ND2"}

pt_shell> get_alternative_lib_cells -current_library n1
{"mylib/ND2P"}

pt_shell> get_lib_cells -of_objects n2
{"libA/ND2"}

pt_shell> get_alternative_lib_cells -current_library n2
{"libA/ND2P"}

pt_shell> size_cell -current_library {U1 U2} ND2P
Information: Sized 'U1' with 'mylib/ND2P'. (NED-045)
Information: Sized 'U2' with 'libA/ND2P'. (NED-045)
1
```

The following example sizes to cell instances to alternative library cells using only specified libraries and the library cell base name. The first library, lib1, is searched first but

it does not contain the target library cell, so the command uses the library cell from the second library.

```
pt_shell> size_cell -libraries {lib1 lib2} {U1 U2} ND2P
Information: Sized 'U1' with 'lib2/ND2P'. (NED-045)
Information: Sized 'U2' with 'lib2/ND2P'. (NED-045)
1
```

See Also

- get_alternative_lib_cells
- report_alternative_lib_cells
- swap_cell
- write_changes
- link_path_per_instance
- link_path

---

sizeof_collection

Returns the number of objects in a collection.

Syntax

int *sizeof_collection* [-categorize]

collection1

Data Types

collection1 collection

Arguments

collection1

Specifies the collection for which to get the number of objects. If the empty collection (empty string) is used for the *collection1* argument, the command returns 0.

-categorize

Return 0, 1, 2 indicating if the collection has 1, or or more than 1 item.

Feedback

Description

The *sizeof_collection* command is an efficient mechanism for determining the number of objects in a collection.

Examples

The following example from PrimeTime shows a simple way to find out how many objects matched a particular pattern and filter in the *get_cells* command.

```
pt_shell> echo "Number of hierarchical cells: \\
?                 [sizeof_collection \\
?                   [filter_collection [get_cells * -hier \\
?                     "is_hierarchical == true"]]]"
Number of hierarchical cells is: 10
```

The following example from PrimeTime shows what happens when the argument for the *sizeof_collection* command results in an empty collection.

```
pt_shell> set s1 [get_cells *]
{"u1", "u2", "u3"}
pt_shell> set ssize [filter_collection $s1 "area < 0"]
pt_shell> echo "Cells with area < 0: [sizeof_collection $ssize]"
Cells with area < 0: 0
pt_shell> unset s1
```

See Also

- collections
- filter_collection

---

snap_objects

Snaps specified objects to the default snap type. An object is snapped onto the nearest position that satisfies the snapping constraint.

Syntax

status *snap_objects*

```
[object_list]
```

Data Types

object_list collection or selection

Feedback

Arguments

object_list

Specifies the collection or selection set containing objects to snap. If this option is not specified, the global selection set is used.

Description

This command snaps the specified objects to the default snap types according to the object types of individual arguments.

The default snap type can be retrieved with the *get_snap_setting* command. Set the snap settings with the *set_snap_setting* command.

Examples

The following example snaps the cell "abc" to the nearest site row slot.

```
prompt> get_snap_setting -class std_cell
row_site
prompt> change_selection [get_cells "abc"]
prompt> snap_objects
```

The alternative syntax uses collection to specify objects.

```
prompt> snap_objects [get_cells "abc"]
```

See Also

- get_selection
- change_selection
- move_objects
- rotate_objects

---

snps.cmd

Python object that provides access to application commands.

Description

Synopsys applications are built around a series of commands. These commands are exposed into the Tcl interpreter and are also available (accessible) from within the Python interpreter as well. In Python these application commands are accessed as methods on the snps.cmd object).

Most application commands accept one or more arguments. The arguments can be either positional or keyword arguments. For example the following command calls the "get_cells"

command to find all soft macros with names matching the glob pattern '*mem*' using Tcl syntax:

```
get_cells *mem* -hierarchical -filter is_soft_macro
```

Using Python syntax, this becomes:

```
cmd.get_cells('*mem*', hierarchical=True, filter='is_soft_macro')
```

The arguments to the various commands are described in the man page for each command. Those may be accessed via the man command. For example:

```
cmd.man('get_attribute')
```

Additionally, Python specific help for a command can be displayed via the Python builtin help facility. This help displays the Python calling syntax. The man pages use Tcl syntax. For example:

```
prompt-py> help(cmd.get_attribute)
get_attribute(...)    Get the value of an attribute
Usage:
    get_attribute(object_spec, attr_name, _class=class_name, quiet=False,
                  value_list=False, objects=object_spec, name=attr_name)

    Positional arguments (brackets indicate optional arguments):
      object_spec:         Object(s) from which to get the attribute
      attr_name:           Name of the attribute

    Keyword arguments (brackets indicate optional arguments):
      [_class]:            If object_spec is a name, this is its class
      [quiet]:             Do not report any messages
      [value_list]:        Force single object return as list
      objects:             Object(s) from which to get the attribute
      name:                Name of the attribute
```

Understanding Manpages

The application man pages for commands describe the Tcl syntax for a command. The Tcl "dash" options map directly to Python keyword arguments. If the dash argument name refers to a Python keyword, then the keyword in Python uses a leading "_" as the above example shows for the "-class" option.

In Tcl, boolean arguments usually accept no value, but in Python all keyword arguments require a value. For boolean argument types you may pass the values: True, 1, False, 0.

Repeating Keyword Arguments

Some of our applications command allow for dash (keyword) arguments to be repeated. This is most common in the timing path related commands. Python does not allow repeating the name of a keyword option. In this case you need to use the "cmd.arg" class

along with the Python keyword expansion syntax (**) to pass the repeated argument names into the command. For example:

```
cmd.get_timing_path(**cmd.arg('through', a, 'through', b))
```

Is equivalent to calling the Tcl command:

```
get_timing_path -through a -through b
```

The "**cmd.arg" syntax can be interspersed with regular keyword argument syntax. Both of these styles are allowed and are equivalent to the first example:

```
cmd.get_timing_path(through=a, **cmd.arg('through', b))
cmd.get_timing_path(**cmd.arg('through', a), through=b)
```

Command Return Values

All commands return a value. The type of the value is one of the following.

Int or Float

Commands returning numeric values return types in one of these Python builtin types.

snps.collection

Commands that return database objects (Tcl collections).

snps.value

All other return types.

Communicating with Tcl

The snps.cmd object can be used to call any command that is registered with the Tcl environment, including Tcl defined procedures (proc). For example:

cmd.set('varName', value)

Sets the Tcl variable 'varName' to the specified value.

cmd.set('varName')

Retrieves the value of the Tcl variable 'varName'.

cmd.source(fileName)

Sources the fileName into the Tcl interpreter.

cmd.eval(commandScript)

Evaluates the supplied Tcl script.

Feedback

See Also

- py_eval

---

snps.collection

Python class for Synopsys collection objects.

Description

Synopsys applications build an internal database of objects and attributes available on those objects. Some examples for class of database objects are designs, libraries, ports, cells, nets, pins, clocks, and so on. Most commands operate on these objects.

Accessing Elements

Collections represent an ordered sequence of database objects. Collection values support the Python sequence interface. The contents of a collection may be accessed like the standard Python list type. For example (x is a collection value):

x[0]

Retrieve the fist item. Returns a single item collection.

x[-1]

Retrieve the last item. All slice based indexing is supported.

len(x)]

Return the number of objects in the collection

for y in x:

Loop over the items. In the loop body "y" is a single item collection.

Building Collections

Collections can be created combining one or more collections. Both operator and function based methods are supported. For example:

x += y # x.append(y)

Add the items in the collection y into the collection referenced by x.

x + y # x.add(y)

Return a new collection by combining the objects in x and y.

x - y # x.remove(y)

Return a new collection containing the objects in x that are not also in y. (x -= y is also supported).

x ^ y # x.remove(y, intersect=True)

Return a new collection that contains the objects that exists in both x and y. (x ^= y is also supported).

Sorting and Filtering

Collections can be sorted and filtered. For example:

x.remove_duplicates()

Returns a new collection with duplicate objects removed.

x.sort(criteria)

Returns a new collection sorted by the sort criteria. This is shorthand for "cmd.sort_collection(x, criteria)"

x.filter(expr)

Returns a new collection by evaluating the filter expression. This is shorthand for "cmd.filter_collection(x, criteria)"

Attribute Values

Collections also provide methods to query and set attribute values on objects. Usually these are used with single item collections (see attribute iterators below for optimized access to attributes on larger collections):

x.set(attr, value)

Calls cmd.set_attribute(x, attr, value)

x.get(attr)

Calls cmd.get_attribute(x, attr)

x.unset(attr)

Calls cmd.remove_attribute(x, attr)

Attribute Iterators

As mentioned above collections may be indexed via an integer or slice value. This gives access to the individual items of a collection. Additionally collections may be indexed by a string or a tuple of strings. Indexing in this way provides access to the attributes of the collection via an iterator. For example:

for name,cost in x['full_name', 'cost']:

Iterate over the values of the full_name and cost attributes in the collection. In the body of the loop 'name' refers to the value of the full_name attribute for the current item.

for obj,(name,cost) in zip(x, x['full_name', 'cost']):

Similar to the above loop, but loop over the collection contents in addition to the attribute values.

x['full_name', 'cost'].to_pandas()

Build a Pandas DataFrame object. The DataFrame contains a column for each attribute specified, with a row for each item in the source collection. See the Pandas documentation for details about the DataFrame type. If the collection is empty, then None is returned.

x['cost'].to_array()

If a single attribute name is used, then the a NumPy array can be requested. If the collection is empty, then None is returned.

DataFrame Contents

A Pandas DataFrame is a grouping of multiple NumPy (np) arrays (each column is an array). When attribute values are converted into these arrays, the type of underlying array differs depending on the values in the attribute. Each attribute type converts like this:

Boolean

The array is of type np.dtype('bool'). If an attribute is unset on an object the value is 'False'.

Int

The array is of type np.dtype('int32') unless an attribute is unset on an object. In that case the type is np.dtype('float64'). Items with the unset attribute have value 'NaN'.

Double

The array is of type np.dtype('float64'). If an attribute is unset on an object, the value is 'NaN'.

Float

The array is of type np.dtype('float32'). If an attribute is unset on an object, the value is 'NaN'.

String

If the attribute values are less than 32 characters and no spaces exist in the in any attribute value (i.e. cannot be interpreted as a Tcl list), then the type is nd.dtype('<UN') where N is the length of the longest attribute result. Otherwise the type is nd.dtype('O') (i.e Object storage) and each entry has the type snps.value. Any unset attribute values are the empty string.

Collection

The array is of type nd.dtype('O') and each entry is of type snps.collection. Unset attributes use the empty collection.

See Also

- py_eval
- collections
- filter_collection

---

snps.redirect

Python class used to redirect application output.

CONSTRUCTOR SYNTAX

```
snps.redirect(file=fileName, compress=False, tee=False)
snps.redirect(stream=ioStream, tee=False)
```

CONSTRUCTOR ARGUMENTS

file

Specifies an output file. This file is created on disk.

stream

Specifies an already open output stream. This can be an already open Python file handle, or any object that supports a write method such as an io.StringIO object.

tee

When True, send output to the specified destination but also send the output to the current output stream.

compress

If the output is to a file, the file is created using zgip compression.

Description

The redirect type is used to redirect tool output to another location. The type is a standard Python execution context object. Output can be redirected into either a file or an in memory buffer stream. This type is used in the Python "with" statement.

Feedback

Examples

```
# Redirect to a compressed file
with snps.redirect('myFile', compresss=True):
  cmd.report_timing()

# Redirect to a variable
out = io.StringIO()
with snps.redirect(out):
  cmd.report_timing()
```

See Also

- py_eval

---

snps.value

Python class for application command return values.

Description

Command return values that are not numeric or collections are returned as values of this type. The snps.value can be treated as either a Python string, sequenced container, or a dictionary as needed.

The documentation for a command will describe the return type.

List Values

For commands that retrieve a list of values, you can treat the returned value as if it is a Python list type. In the following examples, "x" is:

```
x = cmd.get_defined_commands()
```

x[0]

Retrieve the first item (command name). The return value is a snps.value object

x[-1]

Retrieve the last item. All slice based indexing is supported.

len(x)

Return the number of items in the value (number of commands)

for y in x:

Loop over the items. In the loop body "y" is also a snps.value object

Feedback

Dictionary Values

For commands that return key-value pair values, you can treat the returned values as if it is a Python dict type. In the following examples, "x" is

```
x = cmd.get_defined_commands('get_attribute', details=True)
```

x['name']

Access the dictionary key 'name'.

x.keys()

Return an iterator that returns the keys.

x.value()

Return an iterator that returns the values.

for k,v in x.items()

Iterate over the keys and values.

String Values

In many cases the snps.value can also be treated as just a plain string value. However you can explicitly cast a value to a Python str object: str(val)

See Also

- py_eval

---

sort_collection

Sorts a collection based on one or more attributes, resulting in a new, sorted collection. The sort is ascending by default.

Syntax

collection *sort_collection*

```
[-descending]
[-dictionary]
[-limit value_count]
collection
criteria

int value_count
string collection
list criteria
```

Feedback

Arguments

`-descending`

Indicates that the collection is to be sorted in reverse order. By default, the sort proceeds in ascending order.

`-dictionary`

Sort strings dictionary order. For example "a30" would come after "a4".

`-limit` *`value_count`*

Only return the first unique values from the primary sort key.

`collection`

Specifies the collection to be sorted.

`criteria`

Specifies a list of one or more application or user-defined attributes to use as sort keys.

Description

You can use the *sort_collection* command to order the objects in a collection based on one or more attributes. For example, to get a collection of leaf cells increasing alphabetically, followed by hierarchical cells increasing alphabetically, sort the collection of cells using the *is_hierarchical* and *full_name* attributes as *criteria*.

The *criteria* can specify an attribute on another object. The other object must be available as an attribute on this object. For example, if you had a collection of net shapes, you can sort the objects by net using *owner_net.name* for instance.

In an ascending sort, Boolean attributes are sorted with those objects first that have the attribute set to *false*, followed by the objects that have the attribute set to *true*. In the case of a sparse attribute, objects that have the attribute come first, followed by the objects that do not have the attribute.

Sorts are ascending by default. You can specify ascending or descending order on a per attribute basis. You do this by adding a suffix to the attribute name in the sort criteria. The '-' means sort descending and '+' means sort ascending. If no suffix is specified than the default order ascending unless the *-descending* option is used.

The *-limit* option limits the size of the returned collection by choosing the first unique values from the primary sort criteria. For example using a limit of 1 will return all the objects with the smallest (or biggest if descending) value according to the primary sort key.

Examples

The following example from PrimeTime sorts a collection of cells based on hierarchy, and adds a second key to list them alphabetically. In this example, cells i1, i2 and i10

Feedback

are hierarchical, and o1 and o2 are leaf cells. Because the *is_hierarchical* attribute is a Boolean attribute, those objects with the attribute set to *false* are listed first in the sorted collection.

```
pt_shell> set zc [get_cells {o2 i2 o1 i1 i10}]
{"o1" "i2" "o1" "i1" "i10"}
pt_shell> set zsort [sort_collection $zc {is_hierarchical full_name}]
{"o1" "o2","i1" "i10" "i2"}
pt_shell> set zsort [sort_collection -dictionary $zc {is_hierarchical
 full_name}]
{"o1" "o2" "i1" "i2", "i10"}
pt_shell> set zsort [sort_collection -dictionary $zc {area- full_name}]
{"i10" "o1" "o2" "i1" "i2" "i10"}
pt_shell> set zsort [sort_collection -dictionary $zc {area- full_name}
 -limit 1]
{"i10" "o1"}
```

See Also

- collections

---

source

Read a file and evaluate it as a Tcl script.

Syntax

string *source*

```
[-echo]
[-verbose]
[-continue_on_error]
[-sms_scenarios sms_scenarios_list]
file

string file
```

Arguments

`-echo`

Echoes each command as it is executed. Note that this option is a non-standard extension to Tcl.

`-verbose`

Displays the result of each command executed. Note that error messages are displayed regardless. Also note that this option is a non-standard extension to Tcl.

`-continue_on_error`

Don't stop script on errors. Similar to setting the shell variable sh_continue_on_error to true, but only applies to this particular script.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios in which context the script will be read. Using this option is equivalent to using push_sms_scenario before sourcing the script, and pop_sms_scenario after the script.

`file`

Script file to read.

Description

The *source* command takes the contents of the specified *file* and passes it to the command interpreter as a text script. The result of the source command is the result of the last command executed from the file. If an error occurs in evaluating the contents of the script, then the *source* command returns that error. If a return command is invoked from within the file, the remainder of the file is skipped and the source command returns normally with the result from the return command.

By default, source works quietly, like UNIX. It is possible to get various other intermediate information from the source command using the *-echo* and *-verbose* options. The *-echo* option echoes each command as it appears in the script. The *-verbose* option echoes the result of each command after execution.

NOTE: To emulate the behavior of the dc_shell *include* command, use both of these options.

The file name can be a fully expanded file name and can begin with a tilde. Under normal circumstances, the file is searched for based only on what you typed. However, if the system variable sh_source_uses_search_path is set to "true", the file is searched for based on the path established with the search_path variable.

The *source* command supports several file formats. The *file* can be a simple ascii script file, an ascii script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler. Some applications also supported encrupted files. The *-echo* and *-verbose* options are ignored for encrypted formatted files.

Examples

This example reads in a script of aliases:

```
prompt> source -echo aliases.tcl
alias q quit
alias hv {help -verbose}
alias include {source -echo -verbose}
prompt>
```

Feedback

See Also

- search_path
- sh_source_uses_search_path
- sh_continue_on_error

---

starrc_gpd_read_opens_shorts

Creates an error file that contains information in the vicinity of opens and shorts for specified nets or regions.

Syntax

status *starrc_gpd_read_opens_shorts*

```
[-gpd gpd_dir]
[-error_file err_file]
[-windowbbox]
[-type err_type]
[-limit limit]
[-add_gui_selection]
[-add_net_attributes option]
[-nets net_name]
[-summary_view]
[-shorts_types err_type]
```

Data Types

gpd_dir	string
err_file	string
bbox	list
err_type	string
limit	integer
option	string
net_name	string
err_type	string

Arguments

`-gpd gpd_dir`

The GPD generated from the StarRC extraction. The argument is the GPD directory.

`-error_file err_file`

An optional file name for the opens and shorts information; the default is starrc_openshort.err.

`-window`*`bbox`*

The design region to load. If this option is not used, the entire database is loaded. The argument is a list {x1,y1,x2,y2}, where x1 and y1 define the lower-left corner of the region and x2 and y2 define the upper-right corner.

`-type` *`err_type`*

Specifies which errors to analyze. Valid values are *open*, *short*, and *all* (default).

`-limit` *`limit`*

The maximum number of errors to display.

`-add_gui_selection`

If used, nets with opens and shorts are highlighted when the GUI is started. On by default.

`-add_net_attributes` *`option`*

Valid values are *replace* (default) and *append*. If *replace*, creates user attributes starrc_drc_violation and starrc_drc_coordinates. If *append*, previous attributes are not removed.

`-nets` *`net_name`*

A list of one or more nets for which to save extra information around opens and shorts. Glob-style wildcards (*) are supported.

`-summary_view`

Each type of shorts and opens errors gets distinctive color of X marker to categorize the errors.

`-shorts_types` *`err_type`*

Specifies which short error types to analyza. Valid values are net unselectable, nonselected, skip_cell, fill, and blockage.

Then, in the Parasitic Explorer error browser GUI, you can:

- Load types of shorts errors selectively
- Apply types of shorts errors to view the shorts errors
- Load and sort the selected types of shorts errors for debugging

Description

This command creates an error file that contains detailed information in the vicinity of opens and shorts for specified nets or regions.

The sources of information are the GPD and the star directory created after a StarRC extraction run. The Parasitic Explorer error browser then uses the generated error file to display layout information for the selected nets or regions.

Examples

The following example lists only those number of opens errors that fit within the specified design region (bounding box) in the Error Browser GUI:

```
pt_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -window
 0,0,10000,5000 -type open
```

The following example lists only those number of shorts errors that fit within the specified design region (bounding box) in the Error Browser GUI, where the shorts errors types are net *unselectable*, *nonselected*, *blockage*, and *power*:

```
pt_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -window
 0,0,10000,5000 -type short \\
            -short_types (net unselectable nonselected blockage power)
```

The following example lists all shorts errors of the blockage type in the summary view:

```
pt_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -summary_view -type
 short  -short_types blockage
```

The following example lists only that number of shorts errors that fit within the specified design region (bounding box) in the summary view:

```
pt_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -summary_view -type
 short  -window 0,0,10000,5000
```

To view all shorts and opens errors from the actual extracted database, instead of limiting only to the list of specified nets, use the commands as shown in the following example:

```
% StarXtract star_cmd
% StarXtract -write_short_regions -nets_file nets_file star_cmd
 ...
pt_shell> starrc_gpd_read_opens_shorts -gpd <gpd_directory> -error_file
 error_file.txt
```

Before invoking the PrimeTime shell, use the *StarXtract* command to access the original GPD directory and add more nets to see additional errors. Then, use the *starrc_gpd_read_opens_shorts* command to view open and short errors in the Error Browser GUI.

---

start_dsta

This command will begin the HyperGrid distributed analysis.

Feedback

Syntax

status *start_dsta*

```
-script analysis_script
[-num_partitions number_of_partitions]
[-min_partition minimum_number_of_partitions]
```

Data Types

```
analysis_script          string
number_of_partitions     integer
minimum_number_of_partitions     integer
```

Arguments

`-script analysis_script`

Specifies the analysis script to be sourced at manager and worker processes.

`-num_partitions number_of_partitions`

Specifies the desired number of partitions to be started. It is an optional field. If not specified, the tool will automatically infer the number of processes from set_host_options command.

Description

The *start_dsta* begins the HyperGrid distributed analysis for the design. Specifically, it will partition the design at manager and loads partitioned design data at workers. Runs the specified analysis script at manager and worker. The command is supported in *HyperGrid Analysis* mode.

Examples

The following example shows the usage of start_dsta command in *HyperGrid Analysis* mode.

```
pt_shell> start_dsta -script run_basic.pt -num_partitions 2

Information: Starting Hypergrid analysis with 2 partitions.
Start of Manager/Worker Task Processing
----------------------------------------
Started    : Task execution on 'partition0' (Hid=1)
Started    : Task execution on 'partition1' (Hid=2)
Successful : Task execution on 'partition0' (Hid=1)
Successful : Task execution on 'partition1' (Hid=2)
----------------------------------------
End of Manager/Worker Task Processing
```

The number of partitions here is 2 as specified by the user.

```
pt_shell> set_host_options -num_processes 2
pt_shell> start_hosts
pt_shell> start_dsta -script run_basic.pt

Information: Starting Hypergrid analysis with 2 partitions.
Start of Manager/Worker Task Processing
----------------------------------------
Started    : Task execution on 'partition0' (Hid=1)
Started    : Task execution on 'partition1' (Hid=2)
Successful : Task execution on 'partition0' (Hid=1)
Successful : Task execution on 'partition1' (Hid=2)
----------------------------------------
End of Manager/Worker Task Processing
```

The number of partitions here is 2 even though user has not provided -num_partitions option. In such cases, it internally picks the number of processes started by start_host command.

```
pt_shell> start_dsta -script run_basic.pt -num_partitions 4
 -min_partition 2

Information: Starting Hypergrid analysis with 2 partitions.
Start of Manager/Worker Task Processing
----------------------------------------
Started    : Task execution on 'partition0' (Hid=1)
Started    : Task execution on 'partition1' (Hid=2)
Successful : Task execution on 'partition0' (Hid=1)
Successful : Task execution on 'partition1' (Hid=2)
----------------------------------------
End of Manager/Worker Task Processing
```

The number of partitions here is 4 as specified by the user. The minimum number of partitions here is 2 as specified by the user. Here the final number of partitions will be decided based on number of workers available. If number of workers more than or equal to -num_partition then final number of partitions will be num_partition, otherwise final number of partitions will be max(min_partition, number of workers available). Note: -min_partition value can not be less than 1 or more than -num_partition.

See Also

- set_host_options
- start_hosts
- distributed_enable_analysis

Feedback

start_eco_scenarios

Starts running PrimeECO engineering change orders using the hybrid timing view ECO flow.

Syntax

```
status start_eco_scenarios
[-coverage_live_views coverage_limit]
[-include_scenarios scenario_name_list]
[-timeout seconds]
[-type fix_type_list]
```

Data Types

```
coverage_limit       float
scenario_name_list   list
seconds              integer
fix_type_list        list
```

Arguments

`-coverage_live_views coverage_limit`

Specifies the minimum coverage limit to determine live views. For example, if you specify 0.9 for this option, the tool selects live views to achieve 90% live view coverage. Once the number of live views is determined, the tool adjusts host options to launch the same number of worker processes as the live views.

`-include_scenarios scenario_name_list`

Specifies a list of scenarios to include as live views.

Even though PrimeECO selects the best configuration of live and static views, you might want include specific scenarios in live views for various reasons. Use this option to ensure that the specified scenarios are part of the live views selected by the tool. The scenarios specified with this option are always part of the live views, but the tool selects additional scenarios to meet the coverage target.

`-type fix_type_list`

Specifies a list of one or more types of timing violations to fix: *setup*, *hold*, *max_transition*, *max_capacitance*, or *noise*. By default, the tool fixes the *setup*, *hold*, *max_transition*, and *max_capacitance* types.

`-timeout seconds`

Specifies, in seconds, how long the *start_eco_scenarios* command waits for the remote hosts to come online. While the command is waiting, no other commands can be executed at the manager. If all the remote hosts come online before the timeout has expired, the command returns immediately upon

connection of the final host. If all the remote hosts do not come online before the timeout has expired, the command returns and accepts the remainder of the hosts online in the background. The value specified must be greater than or equal to *10s*. If the option is not specified, the default is *21600s* (6 hours).

Description

The *start_eco_scenarios* command runs hybrid timing view ECO operations. In the hybrid timing view ECO flow, the PrimeECO tool uses a combination of live views for accuracy-critical ECO scenarios and static views for less critical scenarios. The static-view timing and violation information is merged into the live-view scenarios at intervals determined by the tool.

The *start_eco_scenarios* command determines live views based on the number of processes specified by the *set_host_options -num_processes* command and launches live views as distributed multi-scenario analysis (DMSA) worker processes. Be sure to specify the number of processes with *set_host_options* before you use the *start_eco_scenarios* command, because this determines the number of live views.

Suppose you have 100 scenarios for your setup and hold timing fixing. If you run ordinary DMSA, you need to have 100 worker processes simultaneously running in one or multiple machines.

In the hybrid timing view ECO flow, if you have 25 live-view scenarios that cover 90 percent of the violations, the tool can merge the remaining 75 scenarios with the 25 live-view scenarios as static views. With this configuration, the tool runs setup and hold fixing with only 25 DMSA worker processes instead of 100.

For another example, because of machine resource limitations, you might be able to run only 10 DMSA worker processes simultaneously. In that case, you can have 10 live views and 90 static views with the live-view timing coverage of 80 percent of the violations. In this configuration, the resource reduction is even greater, but because the live views have less timing coverage, you might not be able to achieve the desired fixing coverage.

Examples

Suppose there are 100 scenarios, S1, S2, ... S100, and each scenario takes 10GB of memory. Running full DMSA requires 1,000 GB of memory to run 100 scenarios. In this example, you want to reduce the memory requirements so that you can run timing fixing on a single machine that has 256 GB of memory.

The following commands explore various configurations of live versus static views to evaluate the memory and coverage tradeoffs.

```
eco_shell> report_eco_scenarios -num_live_views 10
 ...
eco_shell> report_eco_scenarios -num_live_views 20
 ...
```

```
eco_shell> report_eco_scenarios -num_live_views 30
 ...
```

Each report shows the total number of scenarios, the selected number of live ECO scenarios, the number of remaining scenarios to be statically considered, and the resulting live-scenario coverage for the proposed scenario set for each violation type.

The report shows that using 20 live views achieves 95 percent scenario coverage. Because 20 live views requires 200 GB of memory, it satisfies your memory limit of 256 GB.

The following commands start 20 live views.

```
eco_shell> set_host_options -num_processes 20 ...
 ...
eco_shell> start_eco_scenarios
 ...
```

You might want to include specific scenarios in live views to make sure some scenarios are fully covered. The following example shows how to include scenarios S56 and S74.

```
eco_shell> set_host_options -num_processes 20 ...
 ...
eco_shell> start_eco_scenarios -include_scenarios {S56 S74}
 ...
```

The following session explores the usage of different numbers of live views, reporting the timing and other data coverage versus the machine resource usage. Then it performs ECOs using 20 live views running on 20 host processes.

```
create_scenario -name S1 \\
  -specific_data {specific_S2.tcl} -eco_data S1/eco_data
create_scenario -name S2 \\
  -specific_data {specific_S2.tcl} -eco_data S2/eco_data
create_scenario -name S3 \\
  -specific_data {specific_S3.tcl} -eco_data S3/eco_data

report_eco_scenarios -num_live_views 15
report_eco_scenarios -num_live_views 20
report_eco_scenarios -num_live_views 30

set_host_options -num_processes 20
start_eco_scenarios
fix_eco_timing -type setup ...
```

See Also

- add_eco_scenario_data
- create_scenario
- fix_eco_timing

Feedback

- report_eco_scenarios
- set_host_options
- start_hosts
- write_eco_scenario_data

---

start_gui

Starts the application GUI.

Syntax

```
string start_gui
[-file script]
[-no_windows]
[-offscreen 1|0]
[-- x_args ...]

string script
```

Arguments

`-file script`

The given script file is sourced before the GUI starts.

`-no_windows`

The GUI starts without showing the default window.

`-offscreen`

If the specified value is 1 then the GUI starts in offscreen mode.

If the specified value is 0 then the GUI starts in normal (X Display) mode.

`-- x_args`

X11-specific arguments to pass to the X connection.

Description

This command starts the application GUI from the shell prompt. It is ignored if the application GUI has already been started. This is an alias for the gui_start command.

Note the application can only ever connect to one X server during program execution, so changing the value of the DISPLAY environment variable after the first successful start_gui command has no effect.

Examples

The following example starts the application GUI.

shell> *start_gui*

See Also

- stop_gui
- gui_start
- gui_stop

---

start_hosts

Starts the hosts specified by the *set_host_options* command.

Syntax

status *start_hosts*

```
[-timeout   seconds]
[-min_hosts num_hosts]
```

Data Types

```
seconds         integer
num_hosts       integer
host_option_names       list
```

Arguments

`-timeout seconds`

Specifies, in seconds, how long the *start_hosts* command waits for the remote hosts to come online. While the command is waiting, no other commands can be executed at the manager. If all the remote hosts come online before the timeout has expired, the command returns immediately on connection of the final host. If all the remote hosts do not come online before the timeout has expired, the command returns and accept the remainder of the hosts online in the background. The value specified must be greater than or equal to zero. If the option is not specified, the default is 21600s (6 hours).

`-min_hosts num_hosts`

Specifies the minimum number of hosts that the *start_hosts* command waits to come online. While the command is waiting, no other commands can be executed at the manager. As soon as the minimum number of hosts have come online, the command returns immediately and allow commands to execute at the manager. The *-timeout* option to the command overrides the *-min_hosts* option. Therefore, if the minimum number of hosts does not come online before the timeout has expired, the command is governed by the behavior of the *-timeout* option as described previously. The value specified must be greater

than or equal to zero. If the option is not specified, by default the command waits for all hosts launched to come online. In the presence of virtual workers, the *-min_hosts* option refers to the number of real workers. Therefore, *start_hosts* will wait until these many real workers come online along with their respective virtual workers.

host_option_names

Start hosts belonging to the list of host options specified, which were previously defined by the *set_host_options* command. If you do not specify a list of host options, the command starts hosts for all host options by default.

Description

This command explicitly starts the hosts to be used as compute resources. Host options must first be setup using the *set_host_options* command. If the hosts are not explicitly started using this command, they are started during execution of the *update_timing* command.

If the host_options defined by the *set_host_options* command contains virtual workers, then the *start_hosts* command will wait for all workers (including real and virtual) to come online. In this situation however, if the *-min_hosts* option is specified, *start_hosts* will wait until that many real workers come online along with their respective virtual workers.

When hosts are started they end up in one of the following states

```
LAUNCHED      : The process is launched but not ready for use yet.
ONLINE        : The process is online and ready for use.
SHUTDOWN      : The process was shutdown.
FATAL         : The process abnormally terminated.
MISALIGNED    : The manager and worker processes are different versions
                of PrimeTime and the worker process is not usable.
SHUTTING_DOWN : The process is shutting down.
```

Remote Connection Considerations

1. For the manager process to interact with remote processes, there is a connection between the manager and the remote host. Each connection consumes two file descriptors from the operating system. Generally, there is a limit on the number of file descriptors that a user-specified process can open simultaneously. Typically, a user-specified process is allowed to open 1024 simultaneous file descriptors. Because the manager PrimeTime process uses several descriptors for normal operations, a maximum of 500 remote processes can be launched. The *start_hosts* command rejects any hosts beyond 500. To see the system user limit on the number of file descriptors allowed, run the *limit descriptors* command in the UNIX terminal.
2. When using the *rsh* command to launch remote processes, depending on the host configuration, there can be imposted limits in relation to the number of allowed connections.

- The host that you want to connect to can define a limit on the number of simultaneous connections that a process can make to the host in the /etc/xinetd.conf file. The *instances* keyword defines the maximum number of connections allowed. After the number of connections exceeds the limit, the circuit will fail to setup, and a protocol failure will be returned as an error. If this occurs, the manager log file shows the following message:

  poll: protocol failure in circuit setup

  Action: The system limit on the number of incoming connections can be changed only by a superuser. However, you can specify fewer remote processes to be launched on the specific remote host than the instance limit specified by that remote host.

- The host launching the remote processes can define a limit on the range of privileged ports accessible to you. The range is typically 512-1024. After all privileged ports are exhausted, subsequent *rsh* connections will fail to set up, and an out of sockets error will be returned as an error. If this occurs, the manager log file shows the following message:

  rcmd: socket: All ports in use

  Action: The system limit on the number of privileged ports available can be changed only by a superuser. However, you can use a different protocol to launch the subsequent processes beyond the number of privileged ports. For example, *ssh* does not use privileged ports, or you can submit the remote process to a managed resource such as LSF or Grid.

Examples

To start and verify the hosts, use the following flow:

```
# Specify the options of each host
set_host_options -name options1 ...
set_host_options -name options2 ...

# Start the hosts
start_hosts ...

# Verify that the hosts started correctly and report the usage
report_host_usage -verbose
```

See Also

- remove_host_options
- report_host_usage
- set_host_options
- stop_hosts

Feedback

start_learning

Enables PrimeTime to learn users' custom ECO optimization.

Syntax

```
status start_learning
```

Arguments

None

Description

The *start_learning* command enables PrimeTime to learn users' custom ECO optimization. PrimeTime currently supports only the ECO optimization performed by one or more than one *size_cell* commands. After the *start_learning* command is executed, PrimeTime is ready to learn optimization that consists of one or multiple size_cell commands.

After PrimeTime learns custom optimization moves, the *save_training_data* command saves the training data for future usage. Later PrimeTime uses the training data to train itself with the custom optimization moves and apply them to new designs.

Examples

The example scripls below illustrate how to generate custom optimization training data. First perform the *start_learning* command to get PrimeTime ready for custom sizing operations. Then, apply *size_cell* commands manually or source a file that contains multiple *size_cell* commands. This is useful when a customized ECO script generates a sequence of *size_cell* commands. Once sizing operation is completed, use the *save_training_data* command to save the training data for future usage.

```
pt_shell> start_learning
pt_shell> size_cell U100
pt_shell> source my_eco_changes.tcl
pt_shell> size_cell U200
pt_shell> save_training_data -out my_custom_training.td
```

The following example illustrates the *fix_eco_power* command reads the training data in the example above, trains itself with the training data, and reduces power in the current design using the training data.

```
pt_shell> set training_data_directory ./my_training_data_dir
pt_shell> fix_eco_power -training_data my_custom_training.td
```

Feedback

See Also

- fix_eco_power
- save_training_data
- training_data_directory

---

start_multi_user_server

Switches the PrimeTime session into multi-user server mode.

Syntax

status *start_multi_user_server*

```
[-access_group    group_name]
[-max_clients     num_clients]
[-file            script_file]
[-log             log_file]
```

Data Types

```
group_name        string
log_file          string
num_clients       integer
script_file       string
```

Arguments

`-access_group` *group_name*

Specifies the name of the UNIX group used to create the output files written by the multi-user server session. The files written will have read/write permissions enabled for all group members, ensuring that the client users can access and delete their files as needed. A default access_group can be specified by the *multi_user_required_access_group* variable. If both are specified, the option takes precedence.

`-max_clients` *num_clients*

Specifies the maximum number of clients that can connect simultaneously to the current multi-user session. Value must range between 1 <= value <= 16. If this option is not specified, the client limit is either 16 or [current number of cores], whichever is lesser.

`-file` *script_file*

Executes the commands in the specified script file before displaying the multi-user server's pt_shell prompt. If the last statement in the script file is the *quit*,

exit, or *stop_multi_user_server* command, no prompt is displayed and the command shell exits.

`-log` *`log_file`*

Path to file where the logs corresponding to the multi-user session will be written. The verbosity of the logs can be controlled using the *multi_user_logging_level* variable. If the option is not specified, the logs are written in the server's working directory as file named ${server_name}_multi_user.log.

Description

This command configures the PrimeTime session to run in multi-user server mode. The multi-user feature allows different users to connect to a common host and access the same PrimeTime design analysis to run read-only commands in parallel with each other.

The design state cannot be changed after the multi-user session is started. This command performs an implicit timing update if needed.

Upon successful execution, this command starts a new multi-user server shell. The server shell is limited to a subset of PrimeTime commands, including multi-user commands and built-in commands. The set of allowed commands can be queried by running *help ** at the server shell prompt. In the server shell, only PrimeTime variables that do not affect the timing state of the design can be set and modified. If a variable is modified that would change the design timing, a severe error occurs and the multi-user session is terminated.

When this command starts the server session, it reports information about the session, such as: session ID, maximum clients allowed, path to the session logs, verbosity level of session logs, and number of cores available for client use.

If the *start_multi_user_server* command was run in a batch analysis script, analysis script execution pauses while the server session runs.

When the server session is started,

- If a server shell script was specified with the *-file* option of this command, that script is run in the server shell. This script can be used to automate configuration, logging, and maintenance, or to monitor the session or exit the session after a period of time.
- If no server shell script is specified (or the specified script completes), the tool issues an interactive server shell prompt for the duration of the server session.

The server session terminates when the *quit*, *exit*, or *stop_multi_user_server* command is run. Depending on the command,

- If the *stop_multi_user_server* command was used, the original analysis session resumes execution at the point where the *start_multi_user_server* command was run.
- If the *quit* or *exit* command was used, the PrimeTime analysis session exits.

Client Connections

When the *start_multi_user_server* command starts a server session, it prints a report that includes a session ID string. This session ID should be given to any client users who want to connect to the server for analysis.

The server then begins listening on a socket port on the server machine for connection attempts from potential clients. This port number is part of the unique session ID string.

To start a client session, the client user provides the session ID to the *start_multi_user_client* script. When the server receives this request, it forks a subprocess to serve the client analysis requests efficiently.

Data Security

The multi-user analysis feature relies on UNIX groups to control security permissions and protect access to compute hosts and data. It requires a UNIX group to start the session, and it grants access only to client users that belong to that group. It also requires the client user to be logged in (using ssh, rsh, and so on) to the host where the server is running to request access to the session. By leveraging standard and trusted system security protocols (e.g. SSH), the server user can control access to the multi-user analysis by controlling access to the host machine upon which the server is running.

The access group also ensures that all client users have read-write permissions on files written by the client subprocess. Files written by client sessions are stored in the client's working directory, but they are created under the server user's ownership. Read-write permissions are appropriately set for group members, which ensures that client users can access the files and delete them as needed.

Processor Core Management

The maximum number of cores available to client sessions is equal to the number of cores available in the server's PrimeTime process. The server-side analysi script can change the number of cores by running the *set_host_options -max_cores* command before starting the multi-user server session.

The multi-user feature leverages the benefits from Dynamic Core Management (DCM) to optimally distribute cores among active clients. It will allow the client subprocesses to dynamically transfer cores amongst each other depending on the level of multi-threaded algorithm being run on them. Each client sub process will start with one core and DCM will transfer additional cores to clients running MT algorithms if cores are available.

Examples

The following information is printed after running the *start_multi_user_server* command in a PrimeTime session using 6 cores:

```
pt_shell> start_multi_user_server -access_group synopsys -max_client 2
Information: Checked out license 'PrimeTime-ADV-PLUS' (PT-019)
```

```
Information: Switching into multi-user mode. (MC-150)
  Server Name:     user.123
  Session ID:      31:35787
  Server Log:      user.123_multi_user.log
  Logging Level:   low
  Access Group:    synopsys
  Client Cores:    6 (Total)
  Client Limit:    2

pt_shell>
```

You can modify the number of cores allocated to the multi-user session before starting it by using the *set_host_options* command:

```
pt_shell> set_host_options -max_cores 16
  Information: The max_cores limit for the local (current) process has
 been modified by +10 to 16. (CMCR-103)

pt_shell> start_multi_user_server -access_group synopsys -max_client 2
  ...
  Client Cores:    16 (Total)
  Client Limit:    2
  ...
```

See Also

- start_multi_user_client
- report_multi_user_server
- stop_multi_user_server
- stop_multi_user_client

---

start_profile

Starts profiling PrimeTime commands.

Syntax

```
status start_profile
  [-output directory_name]
  [-verbose]
  [-include report_list]
```

Data Types

```
directory_name     string
report_list        list
```

Feedback

Arguments

`-output` *`directory_name`*

Specifies the name of a directory to save the profiling reports to. By default, the output directory name is *profile* in the current working directory.

`-verbose`

Specifies the tracking of all commands in profiling reports. Without this option, only commands which take more than 0.1 second to run or use more than 100 Kb of memory are reported.

`-include` *`report_list`*

Limits the profiling actions to selected items. Allowable list entries are: *stopwatch*, *tcl_profile* and *perf_cpu*.

Description

The *start_profile* command initiates command-level profiling in PrimeTime.

For every PrimeTime command or procedure invocation, the cpu time, elapsed time, delta memory, and number of calls are registered. Several commands or procedure invocations at the same scope level are collapsed into one and their CPU times, elapsed times, and delta memory are added; the number of calls is increased by one.

The PrimeTime profile handles the source command in a special manner. Since source can be used to arbitrarily call deep levels of scripts, source commands also show file names. Therefore, for source commands, even if two source commands are executed at the same scope level, they usually have two different entries, unless they source the exact same file.

In addition to the PrimeTime profile output, a stopwatch report is also created. For each PrimeTime command, a start and an end *events* are recorded. At each stopwatch event, metrics such as current CPU time, elapsed time, and peak memory usage are saved. For each metric, a difference between the current and the previous value is also reported.

PrimeTime profiling reports are written to an output directory at the exit of a PrimeTime session or when the command *stop_profile* is executed. Stopping of profiling also creates a summary HTML page with links to profiling reports or data files with short descriptions.

Use *-include* options to select only the report you want to see. For example, if you require only the code-level performance profiling data, use "*-include {perf_cpu}*".

The *stop_profile* command is used to stop the command level profiling in PrimeTime that was previously started by the *start_profile* command.

In the distributed multi-scenario flow, use *remote_execute* command to activate profiling of remote processes. Profiling is performed on a PrimeTime process, not on a scenario. So if

there are fewer hosts than scenarios, the profiling output appears only in scenario output directories for scenarios that were first to execute on worker hosts.

Examples

In the following script example of profiling, the reports are written to the reports/run_profile directory .

```
start_profile -output reports/run_profile
 ...   # Usual PT script
stop_profile
```

The following script example shows how to enable Tcl profiling in distributed multi-scenario analysis (DMSA). The profiling is first enabled on the DMSA manager and then on DMSA workers.

```
set multi_scenario_working_directory ./ms_data
set manager_profile_dir $multi_scenario_working_directory/manager_profile
start_profile -output $manager_profile_dir
 ...    # Setting up DMSA

current_session -all
remote_execute {
  set outdir
 $multi_scenario_working_directory/[current_scenario]/worker_profile
  start_profile -output $outdir
}
```

See Also

- current_scenario
- remote_execute
- stop_profile

---

stop_gui

Stops the application GUI.

Syntax

string *stop_gui*

Arguments

None.

Feedback

Description

This command stops the application GUI and returns to the shell prompt. It is ignored if the application GUI has not been started or has been stopped. This is an alias for the gui_stop command.

Examples

The following example stops the application GUI and returns to the shell prompt.

shell> *stop_gui*

See Also

- start_gui
- gui_start
- gui_stop

---

stop_hosts

Stops all hosts that have been started.

Syntax

int *stop_hosts*

`[host_option_names]`

Data Types

`host_option_names` `list`

Arguments

`host_option_names`

Shut down all hosts belonging to the list of host options specified, which were previously defined by the *set_host_options* command. If you do not specify a list of host options, the command stops all host options by default.

Description

The *stop_hosts* command shuts down all hosts that have been started. The hosts could have been explicitly launched using the *start_hosts* command or implicitly launched during the execution of the *update_timing* command. Optionally, the *stop_hosts* command shuts down all hosts belonging to a list of host options by specifying the *host_option_names* argument.

In the Distributed Multi-Scenario Analysis (DMSA), stopping hosts will also remove scenarios that can no longer run from the current session. Specifically if a set of scenarios has an affinity with the host options being stopped, all scenarios in the set will be removed.

Examples

In the following example, the manager process is running on the host named ptnlm15 using a 64-bit binary and specifies several different sets of host options.

The first host options, named "my_opts1", specifies

```
- to use one process on the same host as the manager
- no upper limit on the number of cores to use so the default applies.
```

The second host options, named "my_opts2", specifies

```
- to use one process on the same host as the manager, explicitly
  mentioning the manager host by name
- to use a maximum of two cores.
```

The third host options, named "my_opts3", specifies

```
- to use one process on the host named ptnlm11, since no submit
  command is specified, "rsh" will be used to connect to ptnlm11
- to use a maximum of three cores.
```

The fourth host options named "my_opts4", specifies

```
- to use one process on the host named ptnlm12 connecting to
  ptnlm12 using the submit command "/usr/bin/rsh"
- no upper limit on the numebr of cores to use so the default applies.
```

The fifth host options named "my_opts5", specifies

```
- to use one process on an LSF farm, using the "bsub" command found
  from the environment path to submit the job, requiring 1GB of
  memory and two slots per compute server. Notice that the farm job
  must be specified with the number of slots needed to support the
  number of cores to use with the -n 2 -R span\\[ptile=2\\]
  options.
- to use the terminate command "bkill" found from the environment
  path for the termination of jobs that do not come online.
- to use a maximum of two cores.
```

The sixth host options named "my_opts6", specifies

```
- to use one process on a Grid farm, using the "qsub" command found
  from the environment path to submit the job, using the project
  named "bnormal", requiring 1GB of memory and four slots per compute
  server. Notice that the farm job must be specified with the number
  of slots needed to support the number of cores to use by targeting
  the parallel environment "mt" requesting 4 slots with the "-pe mt 4"
  options.
```

```
- to use the terminate command "qdel" found from the environment
  path for the termination of jobs that do not come online.
- to use a maximum of four cores.
```

Note: Before calling the *stop_hosts* command, all the processes report as ONLINE and after calling the *stop_hosts* command all the processes report as SHUTDOWN. The order in which the hosts shutdown is random.

Call the *set_host_options* command to specify the host options.

```
pt_shell> set_host_options -name my_opts1 -num_processes 1
1
pt_shell> set_host_options -name my_opts2 -num_processes 1 \\
          ptnlm15 -max_cores 2
1
pt_shell> set_host_options -name my_opts3 -num_processes 1 \\
          ptnlm11 -max_cores 3
Warning: no submit command specified to access remote host
 'ptnlm10'. The 'rsh' command will be used. (CMCR-004)
1
pt_shell> set_host_options -name my_opts4 -num_processes 1 \\
          -submit_command "/usr/bin/rsh" ptnlm12
1
pt_shell> set_host_options -name my_opts5 -num_processes 1 \\
          -submit_command "[sh which bsub] -n 2 -R rusage\[mem=1000\] \\
          -R span\[ptile=2\]" -terminate_command "[sh which bkill]" \\
          -max_cores 2
1
pt_shell> set_host_options -name my_opts6 -num_processes 1 \\
          -submit_command "[sh which qsub] -P bnormal -pe mt 4 \\
          -l mem_free=1G" -terminate_command "[sh which qdel]" \\
          -max_cores 4
1
```

Call the *start_hosts* command to start the worker processes.

```
pt_shell> start_hosts
Launching 6 Distributed Workers

   1] Launched : /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 4

   2] Launched : ptnlm15 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 2

   3] Launched : rsh ptnlm11 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 3

   4] Launched : /usr/bin/rsh ptnlm12 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 4
```

```
   5] Launched : /lsf/bin/bsub -n 2 -R rusage[mem=1000] -R span[ptile=2]
 \\
                 /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 2

   6] Launched : /remote/sge2/default/bin/lx-amd64/qsub -P bnormal \\
                 -pe mt 4 -l mem_free=1G /J-2014.12/bin/pt_shell \\
                 -slv_type dmsa_slv -max_cores 4


 ------------------------------------------------------------------------
---
  Distributed farm creation timeout : 21600 seconds
  Waiting for   6 (of   6) distributed hosts (Tue Nov 25 04:39:34 2014)
  Waiting for   2 (of   6) distributed hosts (Tue Nov 25 04:39:49 2014)
  Waiting for   0 (of   6) distributed hosts (Tue Nov 25 04:39:59 2014)

 ------------------------------------------------------------------------
---

1
```

Call the *report_host_usage* command after starting the hosts to report the host options specified by the *set_host_options* command, as well as the real time data associated with the processes.

```
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 04:59:46 2014
****************************************

  Options Name          Host Name              Num Processes
  ---------------------------------------------------------------------
  my_opts1              **localhost**          1
  my_opts2              **ptnlm15**            1
  my_opts3              ptnlm11                1
  my_opts4              ptnlm12                1
  my_opts5              >>farm<<               1
  my_opts6              >>farm<<               1

  Options Name     #    Host Name    Job ID    Process ID    Status
  ---------------------------------------------------------------------
  my_opts1         1                 27337     27337         ONLINE
  my_opts2         2    ptnlm15      27378     27378         ONLINE
  my_opts3         3    ptnlm11      23716     23716         ONLINE
  my_opts4         4    ptnlm12      13059     13059         ONLINE
  my_opts5         5    maddog136    312487    21901         ONLINE
  my_opts6         6    peemt32      2165476   5293          ONLINE

  Usage limits (cores)
```

```
  Options Name     #                                                Effective
  ------------------------------------------------------------------------
  (local process)  -                                                4
  my_opts1         1                                                -
  my_opts2         2                                                -
  my_opts3         3                                                -
  my_opts4         4                                                -
  my_opts5         5                                                -
  my_opts6         6                                                -
  ------------------------------------------------------------------------
  Total                                                             4

  Memory usage

  Options Name     #                              Memory (MB)
  ------------------------------------------------------------------------
  (local process)  -                              445.43
  my_opts1         1                              0.00
  my_opts2         2                              0.00
  my_opts3         3                              0.00
  my_opts4         4                              0.00
  my_opts5         5                              0.00
  my_opts6         6                              0.00

  Performance

  Options Name     #    CPU Time (s)              Elapsed Time (s)
  ------------------------------------------------------------------------
  (local process)  -    2                         1023
  my_opts1         1    0                         0
  my_opts2         2    0                         0
  my_opts3         3    0                         0
  my_opts4         4    0                         0
  my_opts5         5    0                         0
  my_opts6         6    0                         0

1
```

Call the *stop_hosts* command with a list of host options to selectively shutdown the processes belonging to the specified host options

```
pt_shell> stop_hosts {my_opts4 my_opts1}
Shutting down worker processes ...
 Shutdown Process 1
 Shutdown Process 4
1
```

Call the *stop_hosts* command to shutdown all active processes.

```
pt_shell> stop_hosts
Shutting down worker processes ...
 Shutdown Process 2
```

```
 Shutdown Process 3
 Shutdown Process 5
 Shutdown Process 6
1
```

Call the *report_host_usage* command after stopping the hosts to report the host options created by the *set_host_options* command.

```
pt_shell> report_host_usage
****************************************
Report : host_usage
Version: J-2014.12
Date   : Tue Nov 25 05:00:00 2014
****************************************

  Options Name          Host Name               Num Processes
  ------------------------------------------------------------------------
  my_opts1              **localhost**           1
  my_opts2              **ptnlm15**             1
  my_opts3              ptnlm11                 1
  my_opts4              ptnlm12                 1
  my_opts5              >>farm<<                1
  my_opts6              >>farm<<                1

  Usage limits (cores)

  Options Name     #                                                Effective
  ------------------------------------------------------------------------
  (local process)  -                                                4
  my_opts1                                                          -
  my_opts2                                                          -
  my_opts3                                                          -
  my_opts4                                                          -
  my_opts5                                                          -
  my_opts6                                                          -
  ------------------------------------------------------------------------
  Total                                                             4

  Memory usage

  Options Name     #                              Memory (MB)
  ------------------------------------------------------------------------
  (local process)  -                              445.43

  Performance

  Options Name     #    CPU Time (s)              Elapsed Time (s)
  ------------------------------------------------------------------------
  (local process)  -    2                         1023

1
```

Feedback

See Also

- start_hosts
- set_host_options
- remove_host_options
- report_host_usage

---

stop_multi_user_client

Terminates a set of specified multi-user client sessions.

Syntax

int *stop_multi_user_client*

```
        client_names_regex
```

Data Types

```
client_names_regex      string
```

Arguments

client_name_regex

Shut down all client sessions whose name matches the specified regular expression pattern. The pattern is a required argument.

Description

The *stop_multi_user_client* command shuts down all active client sessions whose name matches the specified regex pattern. Internally, the command checks the PID of the subprocess serving the specified clients, then sends SIGTERM signals to terminate them.

The specified clients are immediately terminated, even if they are in the middle of processing a command. The client log files will not contain any explicit message indicating that the *stop_multi_user_client* command was the reason for the session termination.

This command reports the names of all clients terminated by the specified pattern.

Examples

The following multi-user server shell command terminates all clients owner by userA:

```
pt_shell> stop_multi_user_client userA*

 Information: Shutting down client 'userA.12785'
```

```
 Information: Shutting down client 'userA.14902'
 Information: Terminated 2 clients.
```

The following command terminates all clients:

```
pt_shell> stop_multi_user_client *

 Information: Shutting down client 'userB.13844'
 Information: Shutting down client 'userC.12158'
 Information: Terminated 2 clients.
```

See Also

- start_multi_user_client
- start_multi_user_server
- report_multi_user_server
- stop_multi_user_server

---

stop_multi_user_server

Stops the currently active multi-user session.

Syntax

int *stop_multi_user_server*

Description

The *stop_multi_user_server* command ends the current multi-user server session. Client users will no longer be able to request access to the PrimeTime analysis using the *start_multi_user_client* script after this command is run.

All currently connected clients are immediately terminated, even if they are in the middle of processing a command. The client log files will not contain any explicit message indicating that the *stop_multi_user_server* command was the reason for the session termination.

After this command is run, the original analysis session (that was in place when the *start_multi_user_server* command was run) is restored, and the timing state of the design can again be modified.

Examples

The following example shows the current multi-user server session being stopped:

```
pt_shell> stop_multi_user_server
1
```

Feedback

See Also

- start_multi_user_client
- start_multi_user_server
- report_multi_user_server
- stop_multi_user_client

---

stop_profile

Stop profiling PrimeTime commands.

Syntax

status *stop_profile*

Arguments

None.

Description

The *stop_profile* command is used to stop the command level profiling in PrimeTime that you started previously using the *start_profile* command. You should use this command in conjunction with the *start_profile* command to write out profiling information for PrimeTime scripts.

Stopping a profile stops collecting information about commands for profiling and writes out profiling reports into the directory specified by *start_profile -output*. By default, the output directory name is *profile* in the current working directory.

This command succeeds all the time except when the profiling is not currently active.

Examples

The following gives an example of profiling.

```
pt_shell> start_profile
pt_shell> ....   # Usual PT script
pt_shell> stop_profile
```

See Also

- start_profile

Feedback

suppress_message

Disables printing of one or more informational or warning messages.

Syntax

string *suppress_message* [*message_list*]

`list message_list`

Arguments

message_list

A list of messages to suppress.

Description

The *suppress_message* command provides a mechanism to disable the printing of messages. You can suppress only informational and warning messages. The result of *suppress_message* is always the empty string.

A given message can be suppressed more than once. So, a message must be unsuppressed (using *unsuppress_message*) as many times as it was suppressed in order for it to be enabled. The *print_suppressed_messages* command displays the currently suppressed messages.

Examples

When the argument to the *unalias* command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to suppress the CMD-029 message:

```
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
prompt> suppress_message CMD-029
prompt> unalias q*
prompt>
```

See Also

- print_suppressed_messages
- unsuppress_message
- get_message_ids
- set_message_info

Feedback

swap_cell

Swaps one or more cells with a new design or library cell.

Syntax

integer *swap_cell*

```
cell_list
swap_in
[-dont_preserve_constraints]
[-file file_name]
[-format db | verilog | vhdl]
```

Data Types

```
cell_list          list
swap_in            string
file_name          string
```

Arguments

cell_list

Specifies a list of cells to be swapped out.

swap_in

Specifies the name of the design or library cell to be swapped in.

-dont_preserve_constraints

Indicates that *swap_cell* is not to reapply the current design constraints after the swap.

-file *file_name*

Specifies the name of a file that contains a design that is to be swapped in.

-format db | verilog | vhdl

Specifies the format for *file_name*. Allowed values are *db* (the default), *verilog*, and *vhdl*.

Description

The *swap_cell* command replaces the cells in the *cell_list* option with *swap_in* (a new design or library cell), and incrementally relinks the design. The command checks to ensure that the pinouts of the swapped-in and swapped-out cells are the same. For example, every boundary pin of the cell being swapped out must have a counterpart with the same name in the cell being swapped in, and vice versa.

The *swap_cell* command is considered a netlist editing command. Like all other netlist editing commands, for the *swap_cell* command to succeed, all of its arguments must

succeed. If any arguments fail, the netlist remains unchanged. The *swap_cell* command returns a 1 if successful and a 0 if unsuccessful.

If you want to swap a library cell for another library cell that is just a different size, use the *size_cell* command instead of the *swap_cell* command. The *size_cell* command is more efficient than the *swap_cell* command, and it is optimized for incremental timing updates.

Library cells being swapped in must be in memory. Designs being swapped in can be in memory or can be retrieved from a file using the *-file* and *-format* options. For more information, see the "Interaction of Reading Netlists and swap_cell" section.

By default, design constraints are preserved by writing a script to the directory referenced by the *pt_tmp_dir* variable, then loading the script after the swap has completed. Use the *-dont_preserve_constraints* option only if you plan to perform several *swap_cell* commands; for example, in a loop. In this case, it is much more efficient to use the *write_script* command then the *swap_cell -dont_preserve_constraints* command, and then the *source* command in your script.

When a subdesign is swapped in or out using swap_cell, the annotated parasitics on nets connected to that cell are removed as they are no longer valid.

If a swapped cell is in the parent chain of any objects that are elements of a saved collection, these collections might be deleted based on the *collection_deletion_effort* variable. For more information, see the man pages for *collections* and associated variables.

The *swap_cell* command does not create black boxes. If the design that is swapped in contains unresolved references, the result of the swap is a partially linked design.

Interaction of Reading Netlists and swap_cell

The best way to swap in a design that is not in memory is to use the *-file* and *-format* options. When you load the new design this way, the *swap_cell* command maintains the current design, and the swap is a one-step operation. Explicitly reading a new design before swapping cells changes the current design, so you must remember to reset the current design back to the previous current design before using the *swap_cell* command. Thus, it is preferable to use the *-file* and *-format* options. Both alternatives are illustrated in the EXAMPLES section.

Interaction of Linking and swap_cell

Do not perform an explicit link after executing *swap_cell*. The *swap_cell* command implicitly relinks the part of the design you have changed; therefore, performing an explicit link after executing *swap_cell* can undo some or all of the work done by *swap_cell*.

For example, consider U2, an instance of design X, which contains instance U0, an instance of design Y. If you swap out U2/U0 for a different design, PrimeTime has modified the *instance* U2/U0; it has not modified the design Y. Therefore, an initial link causes U2/U0 to revert to the original Y (the version of Y inferred by the *link_path*).

PrimeTime does not save information about cells that have been swapped. That information persists only until the next link.

Managing Changes

The changes caused by netlist editing commands are recorded in a *change list*, which is annotated on the design. The contents of the change list can be displayed or exported using the *write_changes* command. The script output formats for PrimeTime and Design Compiler are the most useful. For details, see the *write_changes* man page.

Edits to lower levels of the hierarchy can be undone by swapping in the original version of the design with *swap_cell*. However, edits to the top level of a design cannot be undone without removing the design, rereading the original design, and re linking it.

Uniquification

Uniquification is the process of making each hierarchical block unique in a set of multiple instances of a design. For example, if the block is an instance of BLOCK, PrimeTime searches for a design name (BLOCK_0, BLOCK_1, and so on), until an unused name is found. That becomes the name of the unique design. Uniquification in PrimeTime is somewhat different from uniquification in other Synopsys applications, in that new designs are not created at the time that uniquification occurs. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by *link_design -remove_sub_designs*. Uniquified designs can be listed using *list_designs -all*. They become real designs only when that becomes necessary; for example, when the design is relinked.

The tool automatically uniquifies the necessary blocks -- the edited block and its parent blocks up to the top of the hierarchy. A block that is an instance of a design with multiple instances becomes unique when you edit it. The tool informs you when uniquification occurs.

The following example shows the *size_cell* command being used on a cell and causing uniquification, which is reported by the messages.

```
pt_shell> size_cell n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

In addition to performing automatic uniquification, the tool marks an edited block and its parent blocks with the *is_edited* Boolean attribute, available on a design and hierarchical cells. The following example shows that the value of the *is_edited* attribute on i1/low is *true* after the *size_cell* command was used in the previous example.

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Feedback

Examples

The following example shows the optimal method for swapping in a design from a file. The design "A" is in "A.db". The *swap_cell* command replaces cells u1, u2, and u3 in TOP with the design A (in A.db).

```
pt_shell> current_design
{"TOP"}
pt_shell> swap_cell {u1 u2 u3} A -file A.db
Loading db file '/usr/designs/A.db'
 ...unlinking u1
 ...unlinking u2
 ...unlinking u3
1
pt_shell> current_design
{"TOP"}
```

The following example accomplishes the same task as the previous example, but shows the method where the design file is read explicitly.

```
pt_shell> current_design
{"TOP"}
pt_shell> read_db A.db
Loading db file '/usr/designs/A.db'
1
pt_shell> current_design
{"A"}
pt_shell> current_design TOP
{"TOP"}
pt_shell> swap_cell {u1 u2 u3} A.db:A
 ...unlinking u1
 ...unlinking u2
 ...unlinking u3
1
```

The following example replaces cells u1 and u2 in TOP with the lib_cell misc_cmos/LC3.

```
pt_shell> swap_cell {u1 u2} [get_lib_cells misc_cmos/LC3]
```

See Also

- collections
- current_design
- link_design
- size_cell
- source
- write_changes

Feedback

- write_script
- link_path
- pt_tmp_dir

synchronize_attribute

Synchronizes the setting for an attribute belonging to objects in a collection, across all scenarios in command focus in DMSA mode.

Syntax

status *synchronize_attribute*

```
[-class class_name]
[-must_exist]
[-merge_type rule_name]
object_spec
attribute_name
```

Data Types

```
class_name          string
rule_name           string
object_spec         collection or string
attribute_name      string
```

Arguments

`-class class_name`

Specifies the object class of the objects whose attributes are to be synchronized, in the case where the *object_spec* is a name pattern and not a collection. It can be set to *port*, *cell*, *pin*, or *net*.

`-must_exist`

Unsets a user-defined attribute across all scenarios if the attribute is currently not set in any of these scenarios. By default, the command synchronizes the attributes as specified by the *-merge_type* option (*max* by default) without considering whether the attributes are currently set.

`-merge_type rule_name`

Specifies how to merge different attributes that come from the scenarios, either *min* or *max*. The default is *max*, which uses the maximum numeric value found, or the latest string alphanumerically, or the value *true* for a Boolean attribute. Setting this option to *min* has the opposite merging effect.

object_spec

Specifies the objects on which to set the attribute either as an object collection or a name string combined with the *-class* option. Only a single object specification is accepted.

attr_name

Specifies the name of the attribute to synchronize, either a user-defined attribute or the dont_touch application attribute. Other application attributes cannot be synchronized with this command.

Description

The *synchronize_attribute* command synchronizes the value of an attribute for all objects in a collection, across all scenarios in command focus in DMSA mode. The attribute must be a user-defined attribute or the dont_touch application attribute.

The command sets the specified attribute to the same value across all scenarios, by default choosing the largest numeric value found, or the latest string alphanumerically. For a Boolean attribute, if both of *true* and *false* settings are found, the attributes are synchronized to *true*.

For the opposite synchronization behavior, set the *-merge_type* option to *min*.

Examples

In the following DMSA example, there are multiple scenarios, one of which is called scen1. A script creates a user-defined net attribute and assigns a value to that attribute for all nets, using a different value for scenario scen1. Later, the *synchronize_attribute* command synchronizes the attribute setting to the larger value across all scenarios.

```
pt_shell> remote_execute {
    define_user_attribute -type float -class net attr_float
    if {[current_scenario] == "scen1"} {
      set_user_attribute [get_nets] attr_float 10.0
    } else {
      set_user_attribute [get_nets] attr_float 15.0
    }
  }
  ...
  pt_shell> synchronize_attribute {[get_nets]} attr_float
```

See Also

- collections
- define_user_attribute
- get_attribute

Feedback

t

test_library

Debug command to view library data in the memory.

Syntax

Boolean *test_library*

`-lib_name` *`library_name`*

Data Types

`library_name`	`string`

Arguments

`-lib_name` *`library_name`*

Indicates the library whose data user wants to view.

Description

This command opens an internal shell and user can view various options in that shell to display library data in memory.

u

unalias

Removes one or more aliases.

Syntax

string *unalias*

Feedback

patterns

Arguments

patterns

Specifies the patterns to be matched. This argument can contain more than one pattern. Each pattern can be the name of a specific alias to be removed or a pattern containing the wildcard characters * and %, which match one or more aliases to be removed.

Description

The *unalias* command removes aliases created by the *alias* command.

Examples

The following command removes all aliases.

```
prompt> unalias *
```

The following command removes all aliases beginning with f, and the alias rt100.

```
prompt> unalias f* rt100
```

See Also

- alias

---

undefine_derived_user_attribute

Remove an attribute definition defined in Tcl

Syntax

string *undefine_derived_user_attribute* -classes *class_list*

-name *attribute_name*

```
list class_list
string attribute_name
```

Arguments

-classes class_list

Remove attribute on these classes

-name attribute_name

Name of the attribute to remove

Feedback

Description

The *undefine_derived_user_attribute* command removes an attribute defined by the *define_derived_user_attribute* command.

Examples

To remove the my_attr attribute previously created with the *define_derived_user_attribute* command do the following:

```
prompt> undefine_derived_user_attribute -class cell -name my_attr
```

See Also

- define_derived_user_attribute
- get_attribute

---

undo

Reverses the effects of ECO editing commands (*size_cell*, *insert_buffer*, and *remove_buffer*).

Syntax

```
status undo
[-levels num_levels
[-check_only]
[-silent]
```

Data Types

num_levels integer

Arguments

`-levels num_levels`

Reverses the effects of the specified number of command levels. The default is 1.

`-check_only`

Causes the command to return the same status and messages it would otherwise return without actually changing anything.

`-silent`

Suppresses messages and Tcl errors.

Feedback

Description

This command undoes the effects of one or more ECO editing commands. The command can be used only if enabled in the PrimeTime GUI (ECO > Undo > Enabled). A PrimeTime-ADV-PLUS license is required.

The undone commands can be redone using the *redo* command.

Examples

The following example reverses the effects of a *size_cell* command.

```
pt_shell> size_cell InstSimple/InterNand ND2X4
  ...
pt_shell> undo
  ...
```

See Also

- redo
- undo_config

---

undo_config

Configures the undo stack.

Syntax

```
old_value undo_config
{ -enable | -disable }
```

Arguments

`-enable`

Re-enables the undo stack after it is disabled. See *-disable* option for more details.

`-disable`

Disables the undo stack. The undo is cleared and no new commands are added to the stack until it is enabled.

This option is useful if the user is about to make a large number of changes which he/she knows that they are not going to undo. Disabling in this case will not waste memory and also speed up the command processing.

Description

This command configures the undo stack.

The return value is the old value of the setting.

Examples

The following example disables undo, runs a complex command and re-enables undo:

```
shell> undo_config -disable
shell> fix_eco_drc -type max_fanout
shell> undo_config -enable
```

See Also

- undo
- redo

---

unset_rtl_to_gate_name

Specifies the signals or objects that must never be mapped, or to undo the erroneously mapped signals or objects.

Syntax

status *unset_rtl_to_gate_name*

```
        name
```

Data Types

name string

Arguments

name

Specifies the name of the RTL object that is not changed to the gate-level object by the name-mapping process.

Description

If you have done only the RTL simulation to generate the RTL backward SAIF or RTL VCD file, PrimePower tries to do the name mapping to find the gate-level objects in the RTL SAIF or VCD file. To prevent the erroneous mapping of the RTL-to-gate-level objects, use the *unset_rtl_to_gate_name* command. This command unmaps the erroneous mapping applied to the specified RTL objects.

Feedback

Examples

The following example unmaps the erroneously mapped RTL and gate-level objects. PrimePower searches the *reg_i[1]* object in the gate-level design to see if it appears in the RTL VCD or SAIF file.

```
pt_shell> unset_rtl_to_gate_name reg_i[1]
```

See Also

- set_rtl_to_gate_name

---

unsetenv

Removes a system environment variable.

Syntax

string *getenv*

```
        variable_name
```

Data Types

variable_name string

Arguments

variable_name

Specifies the name of the environment variable to be unset.

Description

The *unsetenv* command searches the system environment for the specified *variable_name* and removes variable from the environment. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the *env* Tcl array variable. The *unsetenv*, commands is a convenience function to interact with this array. It is equivalent to 'unset ::env(*variable_name*)'

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you unset the variable using the *unsetenv* command, you remove the variable value in the application and in any new child processes you initiate from the application using the *exec* command. However, the variable is still set in the parent process.

See the *set* and *unset* commands for information about working with non-environment variables.

Examples

In the following example, *unsetenv* remove the DISPLAY varible from the environment:

```
prompt> getenv DISPLAY
host:0
prompt> unsetenv DISPLAY
prompt> getenv DISPLAY
Error: can't read "::env(DISPLAY)": no such variable
        Use error_info for more info. (CMD-013)
```

See Also

- printenv
- setenv
- getenv

---

unsuppress_message

Enables printing of one or more suppressed informational or suppressed warning messages.

Syntax

string *unsuppress_message* [*messages*]

```
list messages
```

Arguments

messages

A list of messages to enable.

Description

The *unsuppress_message* command provides a mechanism to re-enable the printing of messages which have been suppressed using *suppress_message*. You can suppress only informational and warning messages, so the *unsuppress_message* command is only useful for informational and warning messages. The result of *unsuppress_message* is always the empty string.

You can suppress a given message more than once. So, you must unsuppress a message as many times as it was suppressed in order to enable it. The *print_suppressed_messages* command displays currently suppressed messages.

Feedback

Examples

When the argument to the *unalias* command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to re-enable the suppressed CMD-029 message. Assume that there are no aliases beginning with 'q'.

```
prompt> unalias q*
prompt> unsuppress_message CMD-029
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
```

See Also

- print_suppressed_messages
- suppress_message

---

update_activity

Updates activity information on the current design.

Syntax

```
status update_activity
```

Arguments

None.

Description

Updates activity for the current design without updating power. Activity is also automatically updated by command that retrieves activity results, such as the *get_switching_activity* command and most activity attributes. Note that you must set variable *power_enable_analysis* to TRUE before any power command.

PrimePower can consume either time-based (VCD file) or statistical switching activity information (SAIF file) for activity propagation. For a particular analysis mode, appropriate activity information must be provided. For example, in time_based power analysis mode, a VCD file must be provided. In averaged power analysis mode, any form of switching activity information is accepted. You can specify a VCD file by using command *read_vcd*. The statistical switching activity can be specified by using command *read_saif* or *set_switching_activity*.

In averaged power analysis mode, only averaged activity propagation will be done.

This command can be used when only activities of the design need to be refreshed. For example, this can be used for writing out saif without doing a complete power update. This command runs faster than running update_power for design and is more memory efficient.

Feedback

Examples

This example performs time-based activity propagation for time windows (0-20 and 50-end):

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd -rtl -time {0 20 50 -1} vcd_file.vcd
pt_shell> update_activity
```

This example performs averaged activity propagation:

```
pt_shell> set power_analysis_mode averaged
pt_shell> read_vcd vcd_file.vcd
pt_shell> update_activity
```

See Also

- get_switching_activity
- write_saif
- read_vcd
- report_power_analysis_options
- report_switching_activity
- set_case_analysis
- set_power_analysis_options
- set_switching_activity
- power_analysis_mode
- power_enable_analysis

---

update_budget

Updates budget information on the current design.

Syntax

```
status update_budget
```

Arguments

None.

Description

Updates and compute the budget information for the current design.

Please note that the global budget allocation mode needs to be specified by *set_timing_budget* before issuing this command. pdate_budget *also needs to be explicitly called before report_budget.*

Examples

The following example updates budget information on the current design.

```
pt_shell> update_budget
```

See Also

- set_timing_budget
- report_budget

---

update_metrics

Updates RTL power related metrics on the current design.

Syntax

```
status update_metrics
```

Arguments

None.

Description

Updates RTL power metrics for the current design. This command needs to be called before reporting commands like *report_rtl_metrics*.

Note that you must set variable *power_enable_analysis* to TRUE before any power metrics command.

See Also

- update_power
- report_rtl_metrics

---

update_noise

Performs static crosstalk noise analysis for the current design.

Syntax

int *update_noise*

```
[-full]
```

Arguments

`-full`

By default, the *update_noise* command performs the noise analysis only if the design is not up to date for noise analysis. Using the *-full* option forces the *update_noise* command to perform the noise analysis regardless whether the design is out of date or not.

Description

Updates the crosstalk noise for the current design. The crosstalk noise is also automatically updated by commands or attributes that need the information, such as the *report_timing* command.

The *set_noise_parameters* command controls the settings that are used during the noise analysis update.

The noise analysis is performed at each stage of the design, by calculating the worst case noise bump that aggressors induce on the victim line when the victim line is not switching. Also, if the *-enable_propagation* option is used with the *set_noise_parameters* command, during the noise analysis, calculated noise can also propagate to the next stage and combine with the bumps induced by the aggressors of the next stage.

The induced bumps from the aggressors are calculated based on the characteristics of the driver of the victim net. If the *set_steady_state_resistance* command is specified on the library cell of the driver cell, that information would be used for the calculation of induced bumps from the aggressors. Otherwise, the steady state current-voltage (I-V) characteristics of the library cell of the driver is used. If the I-V information does not exist in the library, the steady state resistance from the library is used. If none of these are available, the steady state resistance of the victim driver is estimated from the delay and slew tables.

By default, the analysis of beyond high and low rails are disabled. If the *-include_beyond_rails* option is used with the *set_noise_parameters* command, noise analysis is also performed for beyond high and low rail during the noise update.

Examples

The following example updates the noise information for the current design.

```
pt_shell> update_noise
```

Feedback

See Also

- report_noise
- set_noise_parameters
- set_steady_state_resistance

---

update_power

Updates power information on the current design.

Syntax

```
status update_power
```

Arguments

None.

Description

Updates power for the current design. Power is also automatically updated by command that retrieves power results, such as the *report_power* command and most power attributes. Command *update_power* explicitly prepares the design for further analysis. Note that you must set variable *power_enable_analysis* to TRUE before any power command.

Starting from 2008.12 release, *update_power* can also perform power waveform generation and peak power calculation.

The *power_analysis_mode* variable can be used to specify the power analysis mode to perform. PrimePower can perform different types of power analysis based on the mode setting. The default power analysis mode is "averaged". For more information, see the *power_analysis_mode* man page.

Command *set_power_analysis_options* can be used to specify the options for power analysis. It must be set before *update_power* to take effects in power analysis. Issuing command *set_power_analysis_options* with no option can reset all the power analysis options to default. Use command *report_power_analysis_options* to get the current analysis option settings. For more information, see the *set_power_analysis_options* man page.

PrimePower can consume either time-based (VCD file) or statistical switching activity information (SAIF file) for power calculation. For a particular analysis mode, appropriate activity information must be provided. For example, in time_based power analysis mode, a VCD file must be provided. In averaged power analysis mode, any form of switching activity information is accepted. You can specify a VCD file by using command

read_vcd. The statistical switching activity can be specified by using command *read_saif* or *set_switching_activity*.

For power estimation using vector free power analysis, all primary inputs and black boxes' pins are assigned with default switching activities. You can use *set_case_analysis* or *set_switching_activity* to improve accuracy. For example, using *set_case_analysis* to set a scan enable pin to 0 gives more accurate average power for normal operating mode. Setting reasonable switching activity values for set/reset pins by using the *set_switching_activity* command is also preferred.

In averaged power analysis mode, only averaged power results will be calculated.

In time_based mode, power waveforms can be generated for the design, specific hierarchies or cells. Power waveforms are series of event-based power data calculated over time while processing VCD activity data. Meanwhile, peak power, which is the largest power value in the waveform, is also captured. The power waveforms are written into waveform files, which can later be displayed by waveform viewing tools. The peak power is saved and can be reported by *report_power*. Besides generating power waveforms, the averaged power results are also calculated.

If a gate-level zero_delay VCD (indicated by *read_vcd -zero_delay*) or a RTL VCD (indicated by *read_vcd -rtl*) is used in time_based mode, cycle accurate peak power analysis will be performed. In cycle accurate peak power analysis, the default sampling interval is the clock cycle associated with the fastest clock in the design. Within the sampling interval the power waveform is a constant which represents the total power dissipated in that cycle. Cycle accurate peak power analysis provides cycle level accurate power behavior for the design. It can report maximum cycle power and the simulation time of the cycle in which the maximum power happens. The results are shown as the peak power reported by *report_power* command and peak power attributes. The *-cycle_accurate_clock* and *-cycle_accurate_cycle_count* options of the *set_power_analysis_options* command can be used to specify the cycle for this analysis.

In leakage-variation power analysis mode, leakage variation analysis will be performed. For more information, see the *power_enable_leakage_variation_analysis* man page.

PrimePower supports UPF domain-based power report feature for multivoltage designs. You can use *set_current_power_domain* or *set_current_power_net* commands to set the domains or supply nets of interest. Only the power dissipated on the specified domains or supply nets are calculated and reported. By default (if not specified), all the domains are included in power analysis. Use *get_current_power_domain* or *get_current_power_net* to show the current settings.

Feedback

Examples

This example performs time-based power analysis for time windows (0-20 and 50-end):

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd -time {0 20 50 -1} dump.vcd
pt_shell> update_power
```

This example performs averaged power analysis:

```
pt_shell> set power_analysis_mode averaged
pt_shell> read_saif -instance top -input file.saif
pt_shell> update_power
```

This example performs time_based power analysis for instance u1.u2 from 100 ns to 500 ns. It also generates the results.out file with time-based power results in .out format.

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd -time {100 500} dump.vcd
pt_shell> set_power_analysis_options -waveform_output results
 -waveform_format out -cells {u1.u2}
pt_shell> update_power
```

The following example performs cycle accurate peak power analysis to generate a cycle accurate waveform by using an RTL VCD file as input:

```
pt_shell> set power_analysis_mode time_based
pt_shell> read_vcd myRTL.vcd -rtl
pt_shell> set_power_analysis_options -waveform_output capp_waveform
 -waveform_format out
pt_shell> update_power
```

See Also

- get_switching_activity
- read_saif
- read_vcd
- report_power
- report_power_analysis_options
- report_switching_activity
- set_case_analysis
- set_current_power_domain
- set_current_power_net
- set_power_analysis_options

Feedback

- set_switching_activity
- write_saif
- power_analysis_mode
- power_enable_analysis

---

update_timing

Updates timing information on the current design.

Syntax

string *update_timing*

```
[-full]
```

Arguments

`-full`

Indicates that the entire timing analysis is to be performed from the beginning. The default is to perform an incremental analysis, which updates only out-of-date information and runs more quickly.

Description

Updates timing for the current design. Timing is also automatically updated by commands that need the information, such as the *report_timing* command. This command explicitly prepares the design for further analysis.

By default, the *update_timing* command uses an efficient timing analysis algorithm that requires minimal computation effort and updates existing timing analysis information only where needed. You can override this default behavior using the *-full* option, which causes the entire timing update to be performed from the beginning. To avoid unnecessarily long run times, use the *-full* option only when you need to override incremental timing analysis. If the design is not timed, the *update_timing* command has the same runtime independent of the *-full* option.

If the *si_enable_analysis* variable is set to *true* and a valid PrimeTime SI license is in place, the *update_timing* command also performs crosstalk-aware timing analysis.

Examples

The following example updates timing information for the current design.

```
pt_shell> update_timing

pt_shell> update_timing -full
```

Feedback

See Also

- report_timing
- si_enable_analysis

---

upf_version

Specify the version for the UPF file and syntax checking.

Syntax

string *upf_version*

`[version_id]`

Data Types

version_id	string

Arguments

version_id

Specifies the UPF version for which the file or UPF commands are intended.

Description

As the UPF standard matures, new updated versions of the UPF standard can occur. The *upf_version* command can be used to specify the version to restrict syntax checking and semantics of UPF commands.

If called with an argument, returns the argument. If called with no argument, returns the current version number.

Currently only version 1.0 is supported. A warning is issued if you specify any other version.

Examples

Here are sample outputs when you use the *upf_version* command to query the current UPF version.

```
pt_shell> upf_version
```

Here is an example of the output that shows all UPF commands, such as commands that might be affected by a specific UPF standard.

```
pt_shell> help UPF
 connect_supply_net    # Connect supply net to supply ports and/or pins
 create_power_domain   # define power supply distribution network
 create_power_switch   # Define a switch in the power domain
```

Feedback

```
create_supply_net     # Create power or ground supply net object
create_supply_port    # Create a port on power domain
get_power_domains     # Create a collection of power domains
....
```

See Also

- create_power_domain
- report_power_domain
- create_supply_net
- create_supply_port

v

validate_ctpm

Validate the accuracy of the CTPM against the golden target library.

Syntax

string *validate_ctpm*

```
-base_lib base_lib
-side_file side_file
-ctpm_file ctpm_file
-golden_lib golden_lib
[-pre_ctpm]
[-pocv]
[-constraint]
[-pin_cap]
[-verbose]
[-verbose_table_as_csv]
```

Data Types

```
base_lib        string
side_file       string
ctpm_file       string
golden_lib     string
```

Arguments

`-base_lib base_lib`

Specifies the base library, which is characterized using original spice model.

`-side_file` *`side_file`*

Specifies the sensitivity augmented library which is the side-file of base library, as used in *define_sensitivity_lib_mapping base_lib -side_file side_file*.

`-ctpm_file` *`ctpm_file`*

Specifies the CTPM file to validate accuracy.

`-golden_lib` *`golden_lib`*

Specifies the golden library, which is characterized using target spice model.

`-pre_ctpm`

Includes pre-CTPM results.

`-constraint`

Includes constraint correlation results.

`-pocv`

Includes parameteric on-chip variation (POCV) correlation results.

`-pin_cap`

Includes pin capacitance correlation results.

`-verbose`

Reports per-lib_timing_arc error details, for both absolute and relative errors. This option applies to both pre-CTPM and post-CTPM reports.

`-verbose_table_as_csv`

Redirects verbose report to csv file(s). pre-CTPM and post-CTPM output will be two seperate files.

Description

The *validate_ctpm* command can be used to validate the acccuracy of PrimeShield SPICE2Design feature at library level. The golden_lib is characterized from target spice model, which can be a spice model representing next PDK, spot model, different spice corner.

Through ML, SPICE2Design aims to bridge the gap between the base and the target by capturing and gap through spice process parameters placed inside of a Compact Timing Power Model (CTPM) database. When CTPM is consumed in a base STA session, the QoR of this session is shifted such that to match QoR of the target.

Examples

The following script validate CTPM accuracy comparing with target library.

Feedback

```
set ps_enable_analysis true
set ps_enable_spice2design_analysis true
set base_lib base_lib.db
set side_file augmented_lib.db
set golden_lib target_golden_lib.db
validate_ctpm -base_lib $base_lib \\
  -side_file $side_file \\
  -golden_lib $golden_lib \\
  -ctpm_file pipeclean.ctpm

1
```

See Also

- gen_ctpm
- read_ctpm
- ps_enable_spice2design_analysis

---

validate_implement_setup

Validates the physical setup specified by the *set_implement_options* command, and checks for consistency between the timing and physical views.

Syntax

```
status validate_implement_setup
  [-capacitance_threshold threshold_value]
  [-resistance_threshold threshold_value]
  [-slack_threshold threshold_value]
```

Data Types

threshold_value	float

Arguments

`-capacitance_threshold` *threshold_value*

Specifies the capacitance difference threshold (in library units) for validating the newly extracted parasitics from StarRC against the existing parastics in the timing sessions. If the difference is greater than this threshold, this command reports it as an error. The default value is 0.01 femto farad.

`-resistance_threshold` *threshold_value*

Specifies the resistance difference threshold (in library units) for validating the newly extracted parasitics from StarRC against the existing parastics in the timing sessions. If the difference is greater than this threshold, this command reports it as an error. The default value is 0.01 ohm.

`-slack_threshold` *`threshold_value`*

Specifies the endpoint slack difference threshold for validating the updated timing slack values (using the newly extracted parasitics) against the slack values in the original timing sessions. If the difference is greater than this threshold, this command reports it as an error. The default value is 1.0 pico second.

Description

The *validate_implement_setup* command validates that both the design and the tool environment are ready to run ECO fixing.

This validation is performed as follows:

- Check that the IC Compiler II and StarRC tools are initialized and ready for work.
- Ensure that the design data (netlist and parasitics) are consistent between PrimeECO and the physical view.
- Identify a fix for a single hold violation.
- Implement the fix in the physical view, then confirm that the fix is legalized and successful.
- Update the design data to reflect the physical changes.
- Confirm that the timing has changed as expected (the insertion endpoints are fixed, other endpoints are negligibly affected).

The *validate_implement_setup* command modifies the design in memory. After you validate your setup, exit the PrimeECO session, then start a new session to fix the full design.

If you encounter an error, correct the error condition and rerun *validate_implement_setup* in a new PrimeECO session.

By default, the command stops if a validation step fails. To force the subsequent validation steps despite a failure, set the *sh_continue_on_error* variable to *true*.

The actual command validation steps are described in the following sections.

Validation step: Initialization setup

This validation step checks if initialization with the IC Compiler II and StarRC tools is successful.

If this step fails, check whether the options specified by the *set_implement_options* command are correct. Possible causes include: incorrect path specification for the IC Compiler II or StarRC executables, incorrect path or contents in the StarRC command file.

If the *set_eco_options* command specifies incorrect physical libraries or physical blocks, the command issues an error.

Validation step: Physical block ECO readiness

This validation step checks if the physical block has any ECO cells that are not in legal locations or have legality violations.

If this validation step fails, it indicates the physical block has ECO cells (such as sized cells or inserted buffers) that have not yet been properly legalized and routed.

Open the block in IC Compiler II and legalize the ECO cells. The physical design must be a legalization and routing clean design before starting the ECO process.

Validation step: Netlist consistency

PrimeECO requires that the netlists be identical between the physical block and timing sessions.

If this validation step fails, it indicates that the netlists are not identical. In this case, PrimeECO reports the netlist differences. Review the logfile to identify the inconsistent cells, then resolve the differences.

Validation step: Parasitics consistency

PrimeECO requires that the StarRC extracted parasitics and the parasitics in the timing sessions are effectively identical. This validation step checks parasitics consistency by removing the existing parasitics, reading the newly extracted parasitics into timing sessions, and comparing the annotated total capacitances and resistances.

If this validation step fails, it indicates a mismatch in parasitics. Possible reasons are:

- StarRC version difference
- StarRC command file difference
- Netlist difference
- Routing topology difference

In this case, PrimeECO reports the nets that have parasitics differences. Review the logfile to identify the inconsistent nets, then resolve the differences.

Validation step: Timing ECO

This validation step checks if a timing ECO can be performed successfully by inserting one buffer to fix a hold violation. (This buffer is also used for the next validation step.)

Validation step: ECO implementation

This validation step checks if the buffer insertion ECO can be successfully implemented. This implementation step includes legalization, ECO routing, parasitics extraction, and timing analysis.

If this validation step fails, it indicates that one of these implementation steps has an issue.

Check if legalization was successful. If it failed, check if any legalization settings are missing in PrimeECO. Specify additional legalization settings with the *-icc2_pre_legalize_script option.*

Check how many cells have moved. PrimeECO expects only a small number of cells to move during the test ECO. If a large number of cells moved, it indicates that the legalization settings might be incorrect or that the physical block is not ready for ECO changes.

Check if ECO routing was successful. If it failed, check if any router settings are missing in PrimeECO. Specify additional router settings with the *-icc2_pre_route_script* option. Check the output of the *check_routes* command before and after ECO routing in the log. Check if the number of physical DRC violations has changed significantly after ECO routing.

Validation step: Final timing QoR consistency

This validation step checks that only expected post-ECO timing changes have occurred.

If this validation step fails, it indicates that large-scale timing changes have occurred, or that timing changes have occurred in unexpected endpoints.

PrimeECO expects that significant timing changes are limited to the endpoints in the fanout of the ECO-changed cells. It also allows for timing changes in other endpoints due to routing changes, but these changes should be minimal due to the small size of the test ECO.

Check the number of cells that moved during legalization. If a large number of cells moved, large-scale timing changes can result. Determine the cause of the cell movements.

Check the number of physical routing DRC violations before ECO routing started. If a large number of physical DRC violations existed before ECO implementation, this can result in large routing changes to fix existing routing violations unrelated to the test ECO.

Examples

The following command runs validation with default options.

```
prompt> validate_implement_setup
```

The following command runs validation with user-specified thresholds.

Feedback

```
prompt> validate_implement_setup \\
                -capacitance_threshold 0.1 \\
                -resistance_threshold 0.1 \\
                -slack_threshold 2.0
```

See Also

- check_eco
- implement_eco
- report_implement_options
- set_implement_options
- set_eco_options

w

which

Locates a file and displays its pathname.

Syntax

string *which filename_list*

`list filename_list`

Arguments

`filename_list`

List of files to locate.

Description

Displays the location of the specified files. This command uses the search_path to find the location of the files. This command can be a useful prelude to read_db or link_design, because it shows how these commands expand filenames. The *which* command can be used to verify that a file exists in the system.

If an absolute pathname is given, the command searches for the file in the given path and returns the full pathname of the file.

Examples

The following examples are based on the following search_path.

prompt> *set search_path "/u/foo /u/foo/test"*

Feedback

The following command searches for the file name foo1 in the search_path.

```
prompt> which foo1
/u/foo/foo1
```

The following command searches for files foo2, foo3.

```
prompt> which {foo2 foo3}
/u/foo/test/foo2 /u/foo/test/foo3
```

The following command returns the full pathname.

```
prompt> which ~/test/designs/sub_design.db
/u/foo/test/designs/sub_design.db
```

See Also

- link_design
- read_db
- search_path

---

win_select_objects

Creates a collection of objects equivalent to a graphical selection operation.

Syntax

string *win_select_objects*

```
[-slct_targets slct_bus]
[-slct_targets_operation operation]
[-create_slct_buses]
[-root instance]
[ -within rectangle | -line line |-at point |
  -radius r | -again_at ]
[-intersect]
[-index i]
[-visible]

string slct_bus
string operation
string instance
```

Arguments

`-slct_targets slct_bus`

Specifies the name of a selection bus to store the result in. By default the result of this command is returned in a collection. If this option is specified selection bus *slct_bus* is used instead. *slct_bus* is also returned as result of the command.

`-slct_targets_operation` *operation*

Specifies which operation should be used when storing the result in selection bus *slct_bus*. This is only allowed if you specify the -slct_targets option. Legal operations are clear, add and remove.

`-create_slct_buses`

Specifies that a new selection bus is created and used to store the result in. Using this option is equivalent to first creating a selection bus using the create_selection_bus command and then providing the created selection bus to option -slct_targets.

`-root` *instance*

Specifies a string for the instance whose component design objects and hierarchy are to be examined against the specified region criteria.

`-within` *rectangle*

Collects design objects lying within the specified rectangle.

`-line` *line*

Collects design objects intersected by specified line.

`-at` *point*

Collects design objects whose bounding box contains the specified point.

`-radius` *r*

Specifies a radius for points specified by the *-at* option. It is applicable only for *_pins* and *_ports*.

`-again_at`

Reapplies the previous *-at* option, but returns a previously-matched design object.

`-index` *i*

For point select (-at option) specifies a specific object index to select. Similar to running -again_at.

`-intersect`

Collects design objects intersecting with specified rectangle. Without this option, the command collects design objects contained in the specified rectangle.

`-visible`

Select objects marked as visible with win_set_select_class command.

Feedback

Description

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

See Also

- win_set_select_class
- win_set_filter

---

win_set_filter

Sets a filter to apply to objects selected by the *win_select_objects* command.

Syntax

```
int win_set_filter
-class class_name
[ -stop_level level ]
[ -start_level level ]
[ -z_level level ]
[ -filter expression ]
[ -layer list ]
[ -user_filter true|false ]
[ -user_filter_cmd tcl_cmd ]
[ -highlighted_only true|false ]
[ -expand_cell_types list ]
[ -part list ]
[ -visible ]
```

Arguments

`-class class_name`

Specifies a single class name for which the filter is to apply.

`-stop_level level`

Specifies the hierarchy level, up to which *win_select_objects* searches for design objects down the hierarchy. This has no effect on objects that are not collectable.

`-start_level level`

Specifies the hierarchy level, from which *win_select_objects* begins searching for design objects down the hierarchy. This has no effect on objects that are not collectable.

`-z_level` *level*

Specifies the interesting Z levels in a 3dic design. The 0 value means all chips in the current 3dic design. Other values mean only the chips on this level and the level below. For example 1 means chips on level 0 and level 1.

`-filter` *expression*

Specifies a filter expression that an object must satisfy to be returned by the *win_select_objects* command. The expression argument must be a Boolean expression based on attributes of the specified class. If this option is not specified, the filter is cleared.

`-layer` *list*

Specifies a layer number list that an object must satisfy to be returned by the *win_select_objects* command. If this option is not specified, the layer filter is cleared.

`-user_filter` *true|false*

Enables user supplied filter command.

`-user_filter_cmd` *tcl_cmd*

Specifies a tcl command for filtering.

`-highlighted_only` *true|false*

Specifies that only highlighted objects can be selected.

`-expand_cell_types` *list*

Specifies cell types for searching design objects down the hierarchy when the search level is greater than 0. This has no effect on types that is not supported or objects that are not collectable.

`-part` *list*

Specifies the object parts to select. When argument is specified only given part of a whole object will be selected by executing *win_select_objects*. The valid values for the *list* argument are *edge*, *vertex*, *centerline*, and *centervertex*.

`-visible`

Specified filters are for visible objects. By default the filters are for selectable objects.

Description

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

Feedback

See Also

- win_select_objects
- win_set_select_class

---

win_set_select_class

Sets the design objects to be collected by the *win_select_objects* command.

Syntax

string *win_set_select_class*

```
{-all | class_names}
[ -visible ]
```

Arguments

`-all`

Selects all classes. The *-all* option and the *class_names* option are mutually exclusive.

`class_names`

Specifies a list of design objects. The *-all* option and the *class_names* option are mutually exclusive.

`-visible`

Specified classes are for visible objects. By default the classes are for selectable objects.

Description

This command is for use by the graphics window(s) for logging interactive, graphical selection operations to the command log for later playback. You are advised not to use this command directly. Use of this command other than by graphics windows may cause unexpected results from selection operations.

See Also

- win_select_objects
- win_set_filter

---

write_activity_fsdb_waveform

Writes gate-level activity waveform for the design netlist in FSDB format.

Feedback

Syntax

status *write_activity_fsdb_waveform*

```
[-file file_name]
[-nets net_list]
[-rtl]
[-enable_power_calculation]
```

Data Types

file_name	string
net_list	list

Arguments

`-file file_name`

Specifies the output waveform file name.

`-nets net_list`

This option allows writing activity waveforms for specified list of nets. By default, PrimePower writes out activity waveform for all logical nets in the deisgns.

`-rtl`

This option allows writing activity waveforms for rtl objects(primary nets and blackbox outputs & sequential outputs) only. By default, PrimePower writes out activity waveform for all logical nets in the deisgns.

`-enable_power_calculation`

This option enables averaged power and activity calculation. By default power calculation is disabled with *write_activity_fsdb_waveform* command.

Description

This command writes the gate-level activity waveforms in FSDB format for whole design netlist. It works only in time-based analysis mode. The command is useful for analysis with RTL or Zero-delay FSDB/VCD. With RTL FSDB/VCD as input, PrimePower would do event based propagation throughout the design and would generate activity waveform for all design nodes.

This command supports delay-aware event propagation with RTL vector and delay-shifting of events in zero-delay event vector as input.

By default, this command only does event propagation and writing of activity waveform for the design netlist. With enable_power_calculation option it also would calculate averaged power and averaged activity for the design, however, peak power calculation would not take place. With this option, user would be able to write out gate-level SAIF file for the design after the command is run successfully.

Feedback

Examples

The following example writes the activity waveform for user specified list of nets.

```
pt_shell> write_activity_fsdb_waveform -nets [get_nets clk*] -file
 clk_nets.fsdb
```

---

write_activity_waveforms

Creates activity waveforms from the Value Change Dump (VCD).

Syntax

integer *write_activity_waveforms*

```
-interval sampling_interval
[-coverage]
[-hierarchical_levels level]
[-exclude_signals signal_list]
[-exclude_cells module_list]
[-output file_name]
[-format fsdb | out]
[-time time_list]
-vcd filename
[-verbose]
[-peak_type min | max]
[-peak_window window_size]
[-strip_path path]
```

Data Types

sampling_interval	float
level	integer
signal_list	list
module_list	list
file_name	string
time_list	list
filename	string
window_size	string
path	string

Arguments

`-interval` *sampling_interval*

Specify the sampling interval that is used for activity waveform. during each interval, activity is aggregated.

`-coverage`

Define activity to be the percentage of nets in a block that toggle at least one time during an interval. By default, the definition of activity is the number of

toggles in the block for the interval, divided by number of nets and the interval period.

`-hierarchical_levels` *level*

Create power waveforms for hierarchical cells only to the specified level from the top. The default is to create waveforms for only the top level of the design.

`-exclude_signals` *signal_list*

This option allows you to exclude some signal names from consideration. The list of signals is a list of signal names. No path should be given with the signal names; all signals with the specified names are ignored from the VCD. This option is useful for excluding signals, such as clock signals from analysis.

`-exclude_cells` *module_list*

This option allows you to exclude some module names from consideration. The list of modules is a list of module names. The names can either exclude a path to the module (such as C instead of TOP/A/B/C), or the full path can be included. If no path is given, all modules with the specified names are ignored from the VCD. The given module names can be globs, such as A*, which would exclude any module starting with 'A'. When a module is excluded, it is not included in the waveforms or the text reports (see the *-verbose* option). Also, the signals in the excluded modules are not considered in the hierarchy above the excluded module. As an exception to this, when a signal is defined in TOP/A/B and also defined in TOP/A/B/C, the signal is still counted as being in B if module C is excluded. The *-exclude_cells* option is useful for excluding macro modules or unsynthesized modules.

The given list of modules needs to be a list of module names in the VCD. It should not be a collection of cells from a design.

`-output` *file_name*

Specify the prefix of the file in which the waveform data is to be written.

`-format fsdb | out`

Specifies the format for the output file. Valid formats are *fsdb* and *out*. The default is *fsdb*.

`-time` *time_list*

Specifies the time windows for the analysis. You might want to only analyze and produce waveforms for a portion of the time in the VCD. The *-time* option accepts a list of numbers as its argument. The argument must have an even number of elements. Each consecutive pair of numbers is a time window; the first number is the start of the window, the second is the end of the window. Activity analysis is only performed on the given time windows.

The given time values are in nanoseconds. The first number in the list can use the special value -1 to represent the beginning of the VCD. The last number in the list can use the special value -1 to represent the end of the VCD.

`-vcd` *`filename`*

Specifies the file name of the VCD to be analyzed. This option is required.

`-verbose`

After analysis, a report is generated, similar to the *report_activity_waveforms* command.

`-peak_type min | max`

Specifies whether the command is to search for an interval with maximum or minimum activity (or coverage, if the *-coverage* option is also specified). By default, the command searches for an interval with maximum activity (or coverage).

`-peak_window` *`window_size`*

Specify the window size to be used when searching for the peak activity period. By default, the peak_window is the same as the interval. PrimePower searches for the peak activity period for each hierarchical block during activity file analysis. The width of the period is the peak_window. For example, the interval might be set to 10 ns, while the peak_window could be 500 ns. In this case, PrimePower looks for activity in a 500 ns window when deciding when the peak activity occurs.

Due to implementation limitations, in the example, the 500 ns window starts and interval boundaries are on. Therefore, the value specified for the *-peak_window* option must be a multiple of the interval value.

`-strip_path` *`path`*

Specifies the path to the module in the VCD of interest. All modules outside the hierarchy under the specified path are excluded. The names in the waveform file are shortened.

Description

Plots the average toggle rate for modules in the VCD over time. The toggle rate during an interval for a module is the average toggle rate for signals in the module for the interval.

No design or library needs to be loaded to use this command; only a VCD is needed.

Activity waveforms can be useful for qualifying the activity vectors. You can review the waveforms to determine if the testbench simulated as expected, and whether the vectors have covered enough of the design to be useful as inputs to power analysis.

The total VCD simulation time is partitioned into intervals, for which the average activity is calculated based on the following formula:

```
# of toggles on all the signals for the interval
Average toggle rate =
 ------------------------------------------------------
                        (# of signals) * (length of interval in
 nanoseconds)
```

You can also view the activity coverage over time to get a sense of how many of the signals are being exercised. A signal is "covered" if it has at least one toggle within the interval, so the coverage per interval is calculated as follows:

```
# of signals with at least one toggle
Coverage = ------------------------------------- * 100%
                     # of signals
```

The output format for the waveforms is either the default of .fsdb or the .out format. View either format with a waveform viewer, such as nWave.

Examples

This example outputs the file#.out text file with waveform information in the .out format:

```
pt_shell> write_activity_waveforms -output file1 -vcd my_vcd.vcd
 -interval 10
```

This example analyzes the time window from 100 to 300 and then from 1000 to the end of the vcd:

```
pt_shell> write_activity_waveforms -output file2 \\
            -vcd my_vcd.vcd -interval 10 -time {100 300 1000 -1}
```

Use the *-exclude_cells* option to exclude some modules from the waveforms and to exclude nets in the modules from being included in the totals for other modules higher in the hierarchy. For example,

```
pt_shell> write_activity_waveforms -output file3 -vcd my_vcd.vcd \\
             -interval 10 -exclude_cells {assertion_module* other_module}
```

See Also

- read_vcd
- report_activity_waveforms

---

write_analytics_data

Write out PrimeTime update_timing runtime and memory information, design parameters and timing analysis settings of the current design in .json format.

Feedback

Syntax

int *write_analytics_data*

```
-tag tag_name
[-output_dir dir_name]
```

Data Types

```
tag_name  string
dir_name  string
```

Arguments

`-tag`

The user-defined file name prefix. It should include design name, scenario name and other data that identify the session. *write_analytics_data* will append timestamp in as part of the file name. Final file name will be <tag>_<timestamp>.json

`[-output_dir`

The optional output repository directory for the output .json file.

Description

The *write_analytics_data* command write out PrimeTime update_timing runtime and memory information, design parameters and timing analysis settings of the current design .json format: <tag>_<timestamp>.json with user-defined file name prefix <tag>. Users can also define the optional output repository directory <dir_name> for the output .json file (if not provided, the default output directory will be the current working directory).

---

write_app_var

Writes a script to set the current variable values.

Syntax

string *write_app_var*

```
-output file
[-all | -only_changed_vars]
[pattern]
```

Data Types

```
file         string
pattern      string
```

Feedback

Arguments

`-output file`

Specifies the file to which to write the script.

`-all`

Writes the default values in addition to the current values of the variables.

`-only_changed_vars`

Writes only the changed variables. This is the default when no options are specified.

pattern

Writes the variables that match the specified *pattern*. The default is "*".

Description

The *write_app_var* command generates a Tcl script to set all application variables to their current values. By default, variables set to their default values are not included in the script. You can force the default values to be included by specifying the *-all* option.

Examples

The following is an example of the *write_app_var* command:

```
prompt> write_app_var -output sh_settings.tcl sh*
```

See Also

- get_app_var
- report_app_var
- set_app_var

---

write_arrival_annotations

Writes arrival and slew annotations for ILMs or contexts of the given list of instances or for all top level instances as a script of commands.

Syntax

status *write_arrival_annotations*

```
[-instances instance_list]
[-context]
[-design]
```

Feedback

Data Types

instance_list list

Arguments

-instances *instance_list*

Writes arrival annotations for the set of pins affecting the given list of instances. The tool looks for appropriate files that contain the list of aggressor driver pins inside the directories meant for these instances, and creates corresponding scripts.

-context

Specifies that the annotations are meant for the context of the block.

-design

Specifies that the annotations are meant for the entire design.

Description

This command writes arrival annotations to be used at block-level or chip-level hierarchical signal integrity analysis. This command is affected by the *pt_ilm_dir* environment variable. The tool looks for directories meant for the list of instances (or all top level instances) at the location given by the *pt_ilm_dir* environment variable. Inside the directory, it looks for wrapper.txt file or ilm.txt file depending on whether the *-context* option is used or not. Then it creates a new file called wrapper.pt.gz or ilm.pt.gz that contains the script needed to annotate the arrivals and slews at block or chip level.

If the annotations are for the chip level and the ILM is being generated for the full design (this happens if the blocks are shielded from each other and the signal integrity context of the instance is not generated), then use the *-design* option. Then, PrimeTime looks for a directory with the name of the block design and looks for ilm.txt file inside the directory.

Examples

The following example writes the annotations for contexts of all top level blocks:

```
pt_shell> write_arrival_annotations -context
```

The following example writes annotations for the ILM of instance I1:

```
pt_shell> write_arrival_annotations -instances {I1}
```

Feedback

See Also

- create_ilm
- create_si_context
- pt_ilm_dir

---

write_binary_aocvm

Creates a binary advanced on-chip variation (AOCV) file from an AOCV file.

Syntax

int *write_binary_aocvm*

```
[-compress]
aocvm_file
binary_file
```

Data Types

```
aocvm_file          string
binary_file         string
```

Arguments

`-compress`

Compresses the output binary AOCV file to reduce file size.

aocvm_file

Specifies the name of the input AOCV file.

binary_file

Specifies the name of the output binary AOCV file.

Description

The *write_binary_aocvm* command creates binary encoded AOCV files from ASCII AOCV files. This command is used to protect sensitive process related information.

The *read_aocvm* command can read binary and compressed binary AOCV files created using the *write_binary_aocvm* command. No additional arguments are required to read binary or compressed binary AOCV files.

An advanced OCV derate table imported from a binary or compressed binary file is not visible in the *report_aocvm* output.

Feedback

Examples

The following example shows how to create a binary advanced OCV file named "binary_file" from an ASCII AOCV file named "aocvm_file".

```
pt_shell> write_binary_aocvm aocvm_file binary_file
```

The following example shows how to create a compressed binary AOCV file named "small_binary_file" from an ASCII AOCV file named "aocvm_file".

```
pt_shell> write_binary_aocvm -compress aocvm_file small_binary_file
```

See Also

- read_aocvm
- remove_aocvm
- report_aocvm

---

write_changes

Writes out netlist changes performed in the current session, such as *size_cell* and *insert_buffer*.

Syntax

status *write_changes*

```
[-format ptsh | text | dctcl | icctcl | icc2tcl | eco | aprtcl]
[-output file_name]
[-reset]
```

Data Types

```
file_name     string
```

Arguments

```
-format ptsh | text | dctcl | icctcl | icc2tcl | eco | aprtcl
```

Specifies one of the following output formats:

- *ptsh* (default) - PrimeTime script
- *text* - List of changes in descriptive text format
- *dctcl* - Tcl script for Design Compiler
- *icctcl* - Tcl script for IC Compiler and IC Compiler II
- *icc2tcl* - Same as *icctcl*

- *eco* - Binary script for the PrimeTime *read_eco_changes* command
- *aprtcl* - Tcl script for third-party place-and-route tool

`-output` *`file_name`*

Writes the change list to the specified file. If you do not use this option, the change list is written to the standard output (console window).

`-reset`

Clears the change list, erasing the history of previous netlist editing commands in the current session. Note that this does not undo the effects of netlist editing; it merely clears out memory of the editing commands.

Description

The *write_changes* command writes out the netlist changes that were made to the current design after the design was linked. The output is typically formatted as a script that can be run in PrimeTime or another tool such as IC Compiler II or a third-party place-and-route tool.

By default, the output goes to the standard output (console window). To write the output to a file, use the *-output* option.

The following commands create entries in the change list:

```
connect_net
create_cell
create_net
disconnect_net
insert_buffer
remove_buffer
remove_cell
remove_coupling_separation
remove_net
rename_cell
rename_net
set_coupling_separation
size_cell
swap_cell
```

As a script, the output of the *write_changes* command is a series of scoping commands (*current_instance*) and netlist editing commands.

The *-format text* option generates one line per atomic editing operation, in order, describing in words the operation, the instance in which it occurred, and the objects involved.

The change list is not necessarily a verbatim log of the actual netlist editing commands run in the session. The change list might be different from the command history in the following situations:

- For objects specified at a higher level of the hierarchy (for example, creating a cell within u1/u2), the change list changes the scope to the lower-level cell using the *current_instance* command and then performs the operation at that level.
- Some object arguments are made explicit by wrapping them within a command that creates a collection, such as *get_pins*.
- Multiple arguments to a command such as *create_cell* are often converted into separate commands.
- Some arguments can be dropped from the change list such as superfluous arguments to the *remove_cell* command.

The *write_changes* command attempts to merge redundant operations to reduce and simplify the change list. For example, if a cell is resized multiple times, it is simplified to a single resize operation. If a buffer is inserted later removed, the associated commands are omitted. Simplification does not occur with the *-format text* option.

This simplification can cause the number of operations in the change list to be less than the number of ECO operations reported by a *fix_eco_** command.

The following commands are not written to the Design Compiler Tcl scripts (*-format dctcl*):

```
remove_coupling_separation
rename_cell
rename_net
set_coupling_separation
swap_cell
```

The following commands are not written to IC Compiler Tcl scripts (*-format icctcl*):

```
create_net -- if the net name contains a wildcard character or hierarchy
 separator
remove_coupling_separation
rename_cell
rename_net
set_coupling_separation
swap_cell
```

The aprtcl Change Format

The *-format aprtcl* option writes the changes as a set of pseudo-Tcl commands that are not specific to a particular tool. These commands can be parsed and incorporated into third-party place-and-route tools. There is no support in PrimeTime to directly replay this script. The change list file contains the list of changes in sequence and in exact order, one single operation will be performed per command.

Feedback

The *aprtcl* format is similar to the *icctcl* format, except for the following commands:

- *insert_buffer* - Insert one buffer or inverter pair per command, shows load pins driven by an inserted buffer or inverter_pair, distinguishes between ABOR and pin buffer

  Pseudo-Tcl command syntax:

  ```
  insert_buffer
    {new_lib_cell}
    [-inverter_pair]
    -new_net_names {net_name}
    -new_cell_names {buff_name}
    [-on_route]
    [-location {x y}]
      | [-location {x1 y1 x2 y2}]
    [-orientation dir]
    [-route_cut_location {x y}]
      | [-route_cut_location {x1 y1 x2 y2}]
    [-target_hier {target_hier_from_top}]
    {pin_or_port_list}
  ```

  - *{new_lib_cell}*: name of new lib_cell to insert
  - *-inverter_pair*: specifies to add inverter pair instead of buffer cell
  - *-new_net_name*: specifies the names of the new nets to add
  - *-new_cell_name*: specifies the names of the new cell to add
  - *-on_route*: specifies on-route insertion
  - *-location*: specifies physical location for a buffer {x y} or inverter pair {x1 y1 x2 y2}
  - *-orientation*: specifies the new orientation, possible values "N, W, S, E, FN, FW, FS, FE"
  - *-route_cut_location*: specifies where to cut the existing route for on-route insertion. For buffer insertion, the format is *{x y}*. For inverter pair insertion, the format is *{x1 y1 x2 y2}*.
  - *{pin_or_port_list}*: list of pins or ports that inserted buffer will drive
  - *-target_hier*: specifies the exact logical hierachy level in which the buffer or inverter pair is inserted. This option is used only in *-on-route* insertions. For details, see SolvNetPlus article 000036348.

- *size_cell* - Resize cell, one resize per command

  Pseudo-Tcl command syntax:

  ```
  size_cell
    {leaf_cell}
    {new_lib_cell}
  ```

```
[-overlap]
[-location {x y}]
[-orientation dir]
[-pin_map {pin_list}]
```

- *{leaf_cell}*: name of existing cell to be resized
- *{new_lib_cell}*: name of new library cell
- *-overlap*: specifies to allow an overlap of the resized cell with existing cells. Legalization will need to adjust the cell later to avoid overlap. Only occupied site mode can allow overlap during resizing.
- *-location*: optional
- *-orientation*: optional
- *-pin_map*: specifies all pin names of a previous and new library cell when the pin name changes due to the size_cell. *-pin_map* is only valid with the *-format aprtcl* option.

- *remove_buffer* - Removes buffer, one remove per command

  Pseudo-Tcl command syntax:

  ```
  remove_buffer {leaf_cell}
  ```

  - *{leaf_cell}*: name of leaf buffer cell to be removed

- *current_instance* - Sets the current hierarchy

  Pseudo-Tcl command syntax:

  ```
  current_instance {hier_name}
  ```

- *create_cell* - Creates a load cell instance in the current design

  Pseudo-Tcl command syntax:

  ```
  create_cell
    {load_cell_name}
    {new_lib_cell}
    [-location {x1 y1}]
    [-orientation dir]
  ```

  - *{load_cell_name}*: instance name of the new load cell created
  - *{new_lib_cell}*: library cell for new cell
  - *-location*: optional
  - *-orientation*: optional

Feedback

- *connect_net* - Connects a net to specified pins or ports

  Pseudo-Tcl command syntax:

  ```
  connect_net
    {net_name}
    {pin_or_port_list}
  ```

  - *{net_name}*: name of the net to connect
  - *{pin_or_port_list}*: list of pins or ports to be connected to the net

  The following commands are not written to *-format aprtcl* scripts:

  ```
  create_net -- if the net name contains a wildcard character or
   hierarchy separator
  remove_coupling_separation
  rename_cell
  rename_net
  set_coupling_separation
  swap_cell
  ```

  The header of the change list file includes a version string (*aprtcl_file_syntax_version*) so that external parsers can adapt to changes in the format over time. Currently, the only possible version value is *1.0*. It is recommended that parser scripts check this version for an expected value, so that future syntax upgrades do not cause unexpected behavior.

Physical ECO

After PrimeTime performs ECOs using physical data, the commands written by *write_changes -format icctcl* contain the placement locations of new buffers.

After you run an ECO using physical data, the tool disables the *ptsh* format. Because PrimeTime keeps track of physical changes internally, do not use the *-reset* option after physical data is loaded into memory. To replay changes in PrimeTime, save the changes using the *write_changes -format eco* command, which writes out the changes in binary format; and read the changes back into a PrimeTime session using the *read_eco_changes* command.

Multiply Instantiated Modules (MIM)

When the *eco_enable_mim* variable is *true*, the *write_changes* command writes one change list for the associated MIM group so that the change list can be directly implemented by the place-and-route tool.

This MIM change-collecting feature is supported by the following changelist formats:

- **`-format icctcl`**
- **`-format aprtcl`**

Feedback

Suppose a CPU module is instantiated four times in module TOP as CPU1, CPU2, CPU3, and CPU4. After fixing is completed with MIM enabled, the following command is used to write change lists:

```
pt_shell> write_changes -format icctcl -output pteco.tcl
```

This creates the following two change lists in the current directory:

```
TOP_pteco.tcl      # A change list for TOP module
CPU_pteco.tcl      # A change list for CPU module
```

The specified output file name is used as postfix after the name of the MIM. Also, the file contains the instance names of the changes to avoid any confusion.

Distributed Multi-Scenario Analysis (DMSA)

The *write_changes* command is compatible with DMSA, with or without netlist data loaded in the manager. It chooses a single scenario and executes the *write_changes* command on the netlist there. It then writes a *single* changelist file to the DMSA working directory. The behavior in DMSA is identical to that seen in a single-scenario run.

Examples

The following session shows some netlist editing commands and the change list created by *write_changes* command. The creation of cells u1 and u2 by a single *create_cell* command is converted to two *create_cell* commands in the change list.

```
pt_shell> create_cell i1/u1 class/AN2
Uniquifying 'i1' (inter) as 'inter_0'.
Information: Created cell 'u1' in 'middle/i1'. (NED-014)
1
pt_shell> size_cell i1/u1 class/AN2P
Information: Sized 'i1/u1' with 'class/AN2P'. (NED-045)
1
pt_shell> create_net i1/net1
Information: Created net 'net1' in 'middle/i1'. (NED-016)
1
pt_shell> connect_net i1/net1 i1/u1/A
Information: Connected 'i1/u1/A' to 'net1'. (NED-018)
1
pt_shell> create_cell {u1 u2} class/AN2
Information: Created cell 'u1' in design 'middle'. (NED-014)
Information: Created cell 'u2' in design 'middle'. (NED-014)
1
pt_shell> write_changes -format ptsh
#####################################
# Change list, formatted for PrimeTime
#####################################
current_instance
current_instance i1
create_cell {u1} {class/AN2}
size_cell {u1} {class/AN2P}
```

```
create_net {net1}
connect_net {net1} [get_pins {u1/A}]
current_instance
create_cell {u1} {class/AN2}
create_cell {u2} {class/AN2}
1

pt_shell> write_changes -format text
Change list, formatted as text:
  1. create_cell in i1: new cell 'u1' lib_cell 'class/AN2'
  2. size_cell in i1: 'u1' sized to 'class/AN2P'
  3. create_net in i1: new net 'net1'
  4. connect_net in i1: pin 'u1/A' to net 'net1'
  5. create_cell in <top>: new cell 'u1' lib_cell 'class/AN2'
  6. create_cell in <top>: new cell 'u2' lib_cell 'class/AN2'
1
```

The following example shows how the change list specifies the location information for IC Compiler.

```
pt_shell> write_changes -format icctcl
########################################
# Change list, formatted for IC Compiler
########################################
current_instance
add_buffer_on_route net1 -user_specified_buffers {U1 BUF1X 102.0 202.0 0}
set_cell_location -coordinates {100.0 200.0} -orientation NF \\
                  [get_cells U1]
insert_buffer U200/Z BUF2X -location {500.0 700.0}
size_cell U100 BUF4X
```

The following example shows how the change list is written for a third-party layout tool.

```
pt_shell> write_changes -format -aprtcl
################################################
# Change list, formatted for Third Party Compiler
# aprtcl_file_syntax_version : 1.0
################################################
current_instance
current_instance {u_dhm_lut}
insert_buffer BUF1X  -new_net_names {net1} -new_cell_names {U1} -on_route
 \\
                     -location {24.6240 26.7520 } -route_cut_location
 {27.512 27.588 } \\
                     -orientation FS { U20/A0 }
insert_buffer BUF1X -new_net_names {net2} -new_cell_names {U2} \\
                    -location {145.6160 112.0240} -orientation N
 { latch/GCLK }
size_cell {cell100} {BUF4X} -location {214.3200 101.9920} -orientation N
size_cell {cell150} {BUF2X} -location {208.2000 115.2300} -orientation S
 -pin_map {{A P} {B Q} {Y Y} }
remove_buffer cell200
current_instance
```

Feedback

```
create_cell {LOAD_CELL} {CLOAD1_LVT} -location {150.6320 61.8640}
 -orientation N
connect_net {u_dhm_unit_0_u_dhm_core_u_convert_64_8_N110} {LOAD_CELL}
```

See Also

- connect_net
- create_cell
- create_net
- current_instance
- disconnect_net
- fix_eco_drc
- fix_eco_power
- fix_eco_timing
- insert_buffer
- remove_buffer
- remove_cell
- remove_coupling_separation
- remove_net
- rename_cell
- rename_net
- set_coupling_separation
- set_eco_options
- size_cell
- swap_cell

write_collection

Output a machine readable report of attribute values for elements in a collection.

Syntax

write_collection

Feedback

```
collection
-file filename]
[-format format]
[-max_rows value_count]
[-columns attribute_list]
[-metadata]

string collection
string filename
string format
int value_count
list attribute_list
```

Arguments

collection

Specifies the collection on which to report. The collection may be homogeneous or heterogeneous. If the collection is very large, you may want to use the *-max_rows* option to limit the size of your output.

`-file` *filename*

This option indicates the name of the file to which the data is written.

`-format` *format*

This option accepts one of the following valid argument values: "csv" | "tsv". If the option is omitted, the default format is "csv", or command separated values. The argument "tsv" produces a tab separated values formatted data.

`-max_rows` *value_count*

This option indicates how many rows of data to include in the output. This number indicates the number of collection elements on which data is output.

`-columns` *attribute_list*

Indicates the set of attributes for which to output values, and their order in the output.

The special attribute name "object_class" may be used to include the element object class names, such as "cell" or "net", in the output.

If the option is omitted, then the object names and object class (or type) names are output by default.

`-metadata`

This option indicates that a meta data section should be included in the output.

Description

This command outputs tabular data of attribute values for elements of the given collection. The attribute values in the output are determined by the list of attributes given to the *-columns* option. The columns in the tabular report will be ordered by the order of attributes in the list. The command *list_attributes* may be called to view a list of attributes defined for an object class.

If the collection is large, the output size may be limited by indicating a *-max_rows* value. The commands *filter_collection* and *sort_collection* may be called to filter and sort a collection based on some criteria prior to creating a report for the collection elements.

By default, the output file omits the meta data section. If *-metadata* option is present, the output initiates with the meta data section which contains information such as the file generation timestamp, the product name and version and column data types. Some consumers of csv/tsv files benefit from the extra information in the meta data while other consumers do not handle it well.

Examples

The following example from IC Compiler II writes the full_name and area values of cells in a collection to a comma separated values (CSV) file named "cell_area.db".

```
icc2_shell> set cells [get_cells U*]
icc2_shell> write_collection $cells -columns {full_name area} \
              -file cell_area_pins.db
```

The next example outputs the same values to a tab separated values (TSV) file.

```
icc2_shell> write_collection $cells -columns {full_name area} \
               -format "tsv" -file cell_area_pins.db
```

The next examples limits the output to a CSV file to the first 10 objects in the collection.

```
icc2_shell> write_collection $cells -columns {full_name area} \
               -file cell_area_pins.db -max_rows 10
```

The next example first sorts the original collection of cells by the area attribute value in descending order, limiting the resulting collection to the top 10 area values. The example then outputs the full_name and area values to a CSV file named top_10_areas.db.

```
icc2_shell set sorted_cells [sort_collection -dictionary \
              $cells {area- full_name} -limit 10]
icc2_shell> write_collection $cells -columns {full_name area} \
              -file top_10_areas.db
```

The next example writes a sorted collection as in the above example but further limits the output to the first 10 objects in the collection. Note that this command may produce an output that is different from the above example. Limiting the sort to the first 10 values may produce a collection with more than 10 objects since multiple objects may share the same area value.

Feedback

```
icc2_shell set sorted_cells [sort_collection -dictionary \
              $cells {area- full_name} -limit 10]
icc2_shell> write_collection $sorted_cells -columns {full_name area} \
              -file first_10_objects_by_area.db -max_rows 10
```

The next example outputs to a CSV file which will include the meta data section.

```
icc2_shell> write_collection $cells -columns {full_name area} \
               -format "csv" -file cell_area_pins.db -metadata
```

See Also

- collections
- list_attributes
- filter_collection
- sort_collection
- report_collection

---

write_context

Writes the timing context information for HyperScale block instances or instances characterized by the *characterize_context* command or *write_hier_data* command, producing a Tcl script or binary-format context information.

Syntax

status *write_context*

```
[-clock]
[-compress]
[-top]
[-output file_directory_name]
[-boundary_only]
[-format ptsh | sdc | dctcl | gbc]
[-exclude_upf]
[-exclude constraints_list]
[-nosplit]
[cell_block_list]
```

Data Types

cell_block_list	list
file_directory_name	string
constraints_list	string

Feedback

Arguments

`-clock`

Writes only clock information of script constraints. By default, all context information is written.

`-compress`

Generates constraint scripts in compressed format.

`-top`

Writes top-level design information only; excludes constraints from all characterized sub-instances.

`-output` *`file_directory_name`*

Specifies the name of the file or directory in which to write the context. A script for a single object is written to a file. Scripts for multiple objects are written to a directory. Binary context information (*-format gbc*) is written to a directory.

`-boundary_only`

Generates a script that includes only the boundary I/O context.

`-format ptsh | sdc | dctcl | gbc`

Specifies the output format for the script. The allowed values are:

ptsh	PrimeTime script format
sdc	Synopsys Design Constraints script format
dctcl	Design Compiler script format
gbc	binary context format

`-exclude_upf`

Excludes UPF information while writing context.

`-exclude` *`constraints_list`*

Excludes the constraint and exception commands specified in the list.

This option is not supported in interactive mode; it must be used in a script.

`-nosplit`

Disables line splitting, resulting in long lines. This option is useful for comparing different versions of generated scripts and for postprocessing the generated script.

`cell_block_list`

Specifies a list of instances or blocks for which the context is written out. If no object is given, the default is to write out all objects that have been characterized

with the *characterize_context* command. For HyperScale blocks, you must explicitly specify the instances; there is no default.

Description

This command writes the timing context of the specified instances or blocks as PrimeTime, SDC, or Design Compiler scripts; or in binary context format. It can write out the context for:

- Block instances characterized by the *characterize_context* command
- HyperScale block instances

You can use the generated context information to set the timing constraints for a block design for synthesis or logic optimization in another tool, or for performing hierarchical timing analysis in the PrimeTime tool.

You can use the command options to specify the type of script to generate and the name of the script file, or specify a directory in which to write out the context information in binary format. If you do not specify an output file or directory, the command writes out the context information to standard output.

To use context information written in script format, source the script using the appropriate tool. To use the context information written in binary format, use the *read_context* command.

Examples

The following example writes to standard output the context information for instances A1 and A2, in Design Compiler format:

```
pt_shell> write_context -format dctcl {A1 A2} -output myDC_dir
```

The following command writes the context information in PrimeTime script format to a file named A1.ptsh in the current directory:

```
pt_shell> write_context -output A1
```

The following example writes all context information for instance A1 in binary context format using the directory named A1BC:

```
pt_shell> write_context -format gbc -output A1BC A1
```

See Also

- characterize_context
- read_context
- report_context

- write_hier_data
- write_physical_annotations
- hier_enable_analysis

---

write_eco_design

Writes an ECO design that consumes less memory to perform ECO than the original design.

Syntax

status *write_eco_design*

```
[-type type_list]
[-endpoints endpoint_list]
[-force]
[-merge_sessions session_list]
[-verbose]
[dir_name]
```

Data Types

type_list	list
endpoint_list	list
session_list	list
dir_name	string

Arguments

`-type type_list`

Writes an ECO design that preserves the specified violation types compared to those in the original design. You can specify one or more of the following violation types:

- *setup* - Preserves setup timing results.
- *hold* - Preserves hold timing results.
- *max_transistion* - Preserves max_transition results.
- *max_capacitance* - Preserves max_capacitance results.
- *max_fanout* - Preserves max_fanout results.
- *noise* - Preserves noise results.
- *timing* - Preserves setup and hold timing results.
- *drc* - Preserves max_transition, max_capacitance, and max fanout results.

If you specify both the *setup* and *hold* violation types (the default behavior), the resulting ECO design will have setup and hold timing reports similar to those of the original design. If you specify only the *hold* violation type, the hold timing report will be similar, but the setup timing report could be different because the ECO design does not contain setup timing information. If you specify more violation types, the ECO design size becomes larger. To minimize the ECO design size, specify only necessary violation types.

`-endpoints` *`endpoint_list`*

Creates an ECO design with the specified timing endpoints. Use this option to customize your ECO design instead of creating an ECO design with violation types. For example, if you have specific timing endpoints for hold fixing but no interest in other violating endpoints, using this option creates an ECO design that preserves hold timing to only these endpoints.

This option provides great flexibility to customize your ECO design. You can write your own scripts to collect specific endpoints based on path groups, timing margins, and slacks. For more examples, see the EXAMPLES section.

`-force`

This options forces to write an ECO design even if ECO information is missing in the saved sessions.

`-merge_sessions` *`session_list`*

Creates an ECO design based on the endpoints merged from the specified PrimeTime saved sessions. Once the option is specified, the *write_eco_design* command merges endpoints from the specified directories and writes an ECO design based on the merged endpoints. Note that the *eco_save_session_data_type* must have been specified before saving the PrimeTime sessions to have violating endpoint data in the saved sessions.

This option is useful when an ECO design is generated without running DMSA, and works only in non-DMSA mode. See the details in the example section.

`-verbose`

Shows additional information such as endpoint names and progress status.

`dir_name`

Specifies a directory name in which an ECO design is created. The specified directory contains multiple files after creation. In Distributed Multi-Scenario Analysis (DMSA), the specified directory has subdirectories in which each scenario creates its own ECO design.

If you do not specify a directory name, the default is *eco_design_directory*.

Feedback

Description

This command reduces hardware requirements during timing closure ECO by creating an ECO design that requires less memory to perform ECO fixing. An *ECO design* is a small subset of the original design that contains only ECO-related design data. For example, it retains only cells and nets on critical paths and omits cells and nets on noncritical paths. Special timing and crosstalk models are created for disconnected nets and pins in the ECO design so that timing quality of results (QoR) in the ECO design is very close to the timing QoR of the original design. Its goal is to have timing QoR close enough to produce similar ECO change lists for the place-and-route tool compared to the change lists generated from the original design.

Note: it is not recommended to signoff with ECO designs. Instead, ECO designs should be used only for ECO purposes, especially when you do not have enough hardware resources to perform timing closure ECO.

Creating an ECO Design With Violation Types

The *write_eco_design* command creates an ECO design that preserves the violation type you specify with the *-type* option. If the *-type* option is not specified, setup and hold will be used as a default. The following command creates an ECO design that preserves setup and hold violations under the default directory, eco_design_directory.

```
write_eco_design
```

The command creates a new directory, eco_design_directory, which contains ECO design data. If a directory with the same name already exists, the command replaces the contents with new data. You can also specify a new directory name if you do not want to use the default name. For example, this command creates an ECO design with the name my_eco_design.

```
write_eco_design my_eco_design
```

After the command creates your new ECO design, use the *read_eco_design* command to load it and run ECO fixing. Suppose your original design consumes 100 gigabytes (GB), but the ECO design consumes only 20 GB. Although you need machines bigger than 100 GB to run static timing analysis (STA), you can perform your ECO on smaller 32-GB machines as the ECO design consumes only 20 GB. For example, after the preceding ECO design is created in the my_eco_design directory, you can log into smaller machines with less memory, run the *read_eco_design* command, and perform timing fixing with the *fix_eco_timing* command:

```
read_eco_design my_eco_design
fix_eco_timing ...
```

Use the *-type* option to specify the violation types that you are interested in to minimize your ECO design size. For example, this command preserves only hold violations:

```
write_eco_design -type hold
```

The resulting ECO design will preserve hold timing violations but might have different other timing violations such as setup and max_transition. However, hold fixing results should be close to the results of the original design because the ECO design has the same critical hold paths preserved.

Specifying more violation types increases the size of an ECO design as it includes more paths. In the following example, the first command creates a bigger ECO design than the second command because it has more violation types.

```
write_eco_design -type {setup hold max_transition max_capacitance}
write_eco_design -type hold
```

Creating an ECO Design in Distributed Multi-Scenario Analysis (DMSA)

In a DMSA flow, run the *write_eco_design* command in the manager to create an ECO design that includes all current scenarios. Suppose you are running DMSA with two scenarios, scen_1 and scen_2. In the following example, a DMSA manager creates an ECO design under the my_dmsa_eco_design directory.

```
write_eco_design my_dmsa_eco_design
```

Under the my_dmsa_eco_design directory, the command creates the scen_1 and scen_2 subdirectories, which contain ECO design data for each scenario. Next, load the ECO design by running the *read_eco_design* command in the manager. The following example launches two worker hosts automatically and loads an ECO design under the my_dmsa_eco_design directory in each scenario. For more DMSA examples, see the man page of the *read_eco_design* command.

```
read_eco_design my_dmsa_eco_design
```

Creating an ECO Design With Saved Sessions

If you perform ECO from PrimeTime sessions that were saved from individual PrimeTime STA runs, follow the steps described in this section to run the *write_eco_design* command efficiently.

First, set the *eco_save_session_data_type* variable to your desired violation types before saving sessions with the *save_session* command in individual PrimeTime STA runs. Setting this variable saves ECO information in addition to timing data while PrimeTime saves its session. The saved ECO information will be used by the *write_eco_design* command to analyze ECO data from different scenarios efficiently. This example script saves setup and hold timing ECO information.

```
set eco_save_session_data_type {setup hold}
save_session scen_1_session
```

Second, start DMSA as you normally do with your DMSA ECO. Since you are running DMSA from individually saved sessions, the following typical DMSA script example

restores saved sessions with the *-image* option of the *create_scenario* command. Suppose there are two saved sessions, scen_1_session and scen_2_session:

```
create_scenario -name scen_1 -image scen_1_session
create_scenario -name scen_2 -image scen_2_session
set_host_options -num_processes 2
start_host
current_session -all
report_global_timing
```

Third, run the *write_eco_design* command in the DMSA manager to create an ECO design that includes both scen_1 and scen_2 scenarios. In the following example, PrimeTime writes an ECO design under the default directory, eco_design_directory:

```
write_eco_design
```

Creating an ECO Design With Fewer Hosts Than Scenarios

You can use fewer hosts than scenarios to create an ECO design. This is especially useful when your machine resource is limited. For example, suppose there are four scenarios that consume 100 GB to run PrimeTime STA for each scenario. However, at the time of ECO, suppose you have only two 100-GB machines available. The following example uses only two machines to create an ECO design from four scenarios:

```
#
# Define four scenarios and their session locations
#
create_scenario -name scen_1 -image scen_1_session
create_scenario -name scen_2 -image scen_2_session
create_scenario -name scen_3 -image scen_3_session
create_scenario -name scen_4 -image scen_4_session

#
# Configure hosts with two machines
#
set_host_options -num_processes 2
start_host

#
# Create an ECO design from four scenarios
#
current_session -all
current_scenario -all
write_eco_design

#
# Exit or additional reporting
# Note that reporting incurs session swapping that may cause runtime
 penalty.
#
# report...
exit
```

It is recommended to have the *write_eco_design* command as the first command right after the *current_session* and *current_scenario* commands before any reporting commands. This avoids scenario swapping that is caused by merging reports from all scenarios.

Creating an ECO Design With Timing Endpoints

Specifying specific endpoint list for the *-endpoints* option enables PrimeTime to create an ECO design that preserves timing to those endpoints. The following example creates an ECO design that preserves timing to FF100/D and FF101/D pins.

```
write_eco_design -endpoints { FF100/D FF101/D }
```

The *-endpoints* option is a powerful tool to customize your ECO design. You can write a Tcl script that collects endpoints you are interested in and specify the endpoints for the option. For example, suppose your design has 100 hold violations, but you are interested in fixing only the 10 worst hold violations. The following example shows how to collect 10 worst hold violations and create an ECO design for them.

```
set paths [get_timing_paths -nw 1 -max 10 -delay min]
set hold_10_endpoints [get_attr $paths endpoint]
write_eco_design -endpoints $hold_10_endpoints
```

The following example creates an ECO design to perform hold fixing only for the *CLOCK_GROUP_A* clock group.

```
set paths [get_timing_paths -group CLOCK_GROUP_A -delay min]
set clk_group_a [get_attr $paths endpoint]
write_eco_design -endpoints $clk_group_a
```

Memory Reduction

After an ECO design is written, memory reduction may not be proportional to the number of reduced cells because PrimeTime's memory consumption also includes fixed memory overhead like loading library. For example, a design loaded with a large number of larger libraries shows lower memory reduction than a design loaded with one small library.

PrimeTime shows an estimated design reduction summary after the *write_eco_design* command finishes. Note that the numbers are not exact but estimated values. The purpose of the estimate is to provide rough numbers for design reduction. Exact cell, net, and pin counts as well as peak memory will be available when the *read_eco_design* command finishes.

Optimizing disk usage

If saved sessions are kept only for ECO purpose, you can reduce disk usage by removing the saved sessions after creating an ECO design. This is especially useful when there are many scenarios that take large disk space.

Creating an ECO design in non-DMSA mode

Specifying saved session directories for the *-merge_sessions* option in non-DMSA mode enables PrimeTime to merge the violating endpoints from the saved sessions and create an ECO design based on the merged endpoints. This option is designed to support to generate an ECO design without running DMSA.

Suppose there are two STA scripts as shown below.

```
# Script STA for FF running in Host1
read_verilog...
update_timing...
set eco_save_session_data_type timing
save_session $my_sessions/STA_SESSION_FF

# Script STA for SS running in Host2
read_verilog...
update_timing...
set eco_save_session_data_type timing
save_session $my_sessions/STA_SESSION_SS
```

Note that the *eco_save_session_data_type* variable is specified to store violating endpoint data in the saved sessions. After the saved sessions are created and available, the following scripts can be used to generate an ECO design under $rreco_db directory.

```
# Script to write ECO design for FF running in Host1
restore_session $my_sessions/STA_SESSIONS_FF
set sess_list "$my_sessions/STA_SESSION_FF $my_sessions/STA_SESSION_SS"
write_eco_design $rreco_db/FF -merge_sessions $sess_list

# Script to write ECO design for SS running in Host2
restore_session $my_sessions/STA_SESSIONS_SS
set sess_list "$my_sessions/STA_SESSION_FF $my_sessions/STA_SESSION_SS"
write_eco_design $rreco_db/SS -merge_sessions $sess_list
```

Note that the *write_eco_design* command specifies a subdirectory under $rreco_db so that each subdirectory is assigned to each scenario. In this directory structure, the following *read_eco_design* command can read in the ECO design in DMSA mode.

```
# Script to read RRECO design in DMSA mode
read_eco_design $rreco_db
```

Examples

This section describes a few examples to convert existing ECO scripts to new scripts utilizing the *write_eco_design* and *read_eco_design* commands.

Single-Scenario Example Scripts

The following example is a typical script for STA and ECO combined together.

```
##################################################
# FILE: run_sta_eco.tcl
#   Typical script for STA and ECO in PrimeTime.
#   Runs in 100 GB machine.
##################################################
#
# STA
#
read_verilog...
read_parasitics...
read_sdf...
update_timing...
report_timing...
#
# ECO
#
set_eco_options...
fix_eco_timing...
write_changes...
exit
```

The preceding script is separated into the two following scripts. The first script creates an ECO design after STA, and the second script loads the ECO design and performs ECO. The second script runs on smaller machines assuming peak memory is reduced from 100 GB to 20 GB.

```
##################################################
# FILE: run_sta_and_write_eco_design.tcl
#   To run STA and write_eco_design.
#   Runs in 100 GB machine.
##################################################
#
# STA
#
read_verilog...
read_parasitics...
read_sdf...
update_timing...
report_timing...
#
# Creates an ECO design
#
write_eco_design
exit

##################################################
# FILE: run_read_eco_design_and_eco.tcl
#   To run ECO.
#   Runs in 20 GB machine.
##################################################
#
# Read the ECO design
```

```
#
read_eco_design
#
# ECO
#
set_eco_options...
fix_eco_timing...
write_changes...
exit
```

Multi-Scenario Example Scripts

The following example is a typical script for DMSA ECO that restores saved sessions and performs ECO.

```
##########################################################################
# FILE: run_dmsa_eco.tcl
#   Typical script for DMSA ECO running from saved sessions.
#   Runs in four 100 GB machine assuming each scenario takes 100 GB.
##########################################################################
#
# Define four scenarios and their session locations
#
create_scenario -name scen_1 -image scen_1_session
create_scenario -name scen_2 -image scen_2_session
create_scenario -name scen_3 -image scen_3_session
create_scenario -name scen_4 -image scen_4_session
#
# Configure hosts with two machines
#
set_host_options -num_processes 4
start_host
#
# Create an ECO design from four scenarios
#
current_session -all
current_scenario -all
#
# ECO
#
fix_eco_timing...
remote_exec { write_changes... }
exit
```

The preceding script is separated into the two following scripts. The first script creates an ECO design with only one 100-GB machine, and the second script loads the ECO design and performs ECO. Assuming each scenario is reduced from 100 GB to 20 GB, all four scenarios can run in a single 100-GB machine.

The first script to create an ECO design in one host is shown below.

```
##########################################################################
# FILE: run_write_eco_design_in_one_host.tcl
#   Creates an ECO design in DMSA four scenarios with one host.
#   Runs in one 100 GB machine assuming each scenario takes 100 GB.
##########################################################################
#
# Define four scenarios and their session locations
#
create_scenario -name scen_1 -image scen_1_session
create_scenario -name scen_2 -image scen_2_session
create_scenario -name scen_3 -image scen_3_session
create_scenario -name scen_4 -image scen_4_session
#
# Configure one host
#
set_host_options -num_processes 1
start_host
#
# Create an ECO design from four scenarios
#
current_session -all
write_eco_design my_dmsa_eco_design
exit
```

The second script to run four scenarios in one 100 GB machine is shown below.

```
##########################################################################
# FILE: run_read_eco_design_and_eco_in_one_host.tcl
#   Load the ECO design with DMSA four scenarios and performs ECO.
#   Runs in one 100 GB machine running four scenarios simultaneously.
#   Assumes each scenario takes 20 GB.
##########################################################################
#
# Load the ECO design
#
read_eco_design my_dmsa_eco_design
#
# ECO
#
fix_eco_timing...
remote_exec { write_changes... }
exit
```

See Also

- read_eco_design
- eco_save_session_data_type

Feedback

write_eco_scenario_data

Writes out scenario timing data needed for ECO fixing in preparation for hybrid timing view ECO fixing.

Syntax

```
status write_eco_scenario_data
-output_directory directory_name
[-group group_list]
[-type fix_type_list]
```

Data Types

```
directory_name        string
group_list            list
fix_type_list         list
```

Arguments

`-output_directory directory_name`

Specifies the directory that will store the necessary timing information of the current scenario for hybrid timing view ECO fixing.

`-group group_list`

Writes timing information only for the specified path groups. This option applies only to the data written for the *setup* and *hold* fixing types.

`-type fix_type_list`

Specifies a list of one or more types of timing violations to fix. The valid values are *setup*, *hold*, *max_transition*, *max_capacitance*, and *noise*.

The default types are always written in the ECO data. The default list is *setup*, *hold*, *max_transition*, and *max_capacitance*. Use the *-type* option to add *noise* to this list.

Description

The *write_eco_scenario_data* command writes out all necessary timing information of the current scenario into a specified directory. This information is used in hybrid timing view ECO fixing.

In the hybrid timing view ECO flow, the PrimeECO tool uses a combination of live views for accuracy-critical ECO scenarios and static views of less critical scenarios. The static-view timing and violation information is merged into the live-view scenarios at intervals determined by the tool.

Based upon the reference information across all scenarios written by the *write_eco_scenario_data* command, the hybrid timing view ECO core engine optimizes

the combination of live views and static views, and delivers the best ECO fixing QoR using fewer machine resources.

Distributed Multi-Scenario Analysis (DMSA)

The *write_eco_scenario_data* command writes out data only for the current scenario.

Examples

The following example writes hybrid timing view ECO data for all scenarios from the DMSA manager into a directory named eco_data.

```
pt_shell> remote_execute {write_eco_scenario_data \\
            -output_directory $sh_launch_dir/eco_data/[current_scenario]}
```

The following example writes hybrid timing view ECO data including noise.

```
pt_shell> write_eco_scenario_data -output_directory S1/eco_data \\
            -type {setup hold max_transition max_capacitance noise}
```

The following session example shows how to generate hybrid timing view data for all scenarios and use that information for hybrid timing view ECO fixing.

```
create_scenario -name S1 \\
  -specific_data {specific_S2.tcl} -eco_data S1/eco_data
create_scenario -name S2 \\
  -specific_data {specific_S2.tcl} -eco_data S2/eco_data
create_scenario -name S3 \\
  -specific_data {specific_S3.tcl} -eco_data S3/eco_data

report_eco_scenarios -num_live_views 15
report_eco_scenarios -num_live_views 20
report_eco_scenarios -num_live_views 30

set_host_options -num_processes 20
start_eco_scenarios
fix_eco_timing -type setup ...
```

For scenarios S1, S2, and S3, the *write_eco_scenario_data* command generates the hybrid timing view data in directories named S1/eco_data, S2/eco_data, and S3/eco_data. This preparation step is necessary for the hybrid timing view ECO flow.

See Also

- create_scenario
- fix_eco_timing
- report_eco_scenarios
- start_eco_scenarios

Feedback

write_eco_session

Writes PrimeTime timing and ECO data to a session, which can be read by the PrimeClosure tool to perform ECO.

Syntax

```
status write_eco_session
[-include session_data_type_list]
[-data_types fix_type_list]
[-data_format ascii | binary]
[-leakage_scenario scenario_name]
[-dynamic_scenario scenario_name]
[-pba_mode none | path | exhaustive | ml_exhaustive]
[-pt_version specific | compatible ]
[-link_session dir_name]
[-spef_data list_of_spef_files]
[-netlist_data list_of_netlist_files]
directory dir_name
```

Data Types

session_data_type_list	list
fix_type_list	list
scenario_name	list
list_of_spef_files	list
list_of_netlist_files	list
pba_mode_value	string
version	string
dir_name	string

Arguments

`-include` *session_data_type_list*

Specifies the session content types to include. Session content types are one or more of the following: pt_session, eco_data, design, netlist, spef, gpd, or libraries. The default types are pt_session and eco_data. Specifying the *design* type is the same as specifying both the netlist and SPEF design types. Specifying the *libraries* type includes libraries in the pt_session, which is equivalent to PrimeTime's *save_session* command with the *-include libraries* option specified.

`-data_types` *fix_type_list*

Specifies one or more fixing data types of setup, hold, max_transition, max_capacitance, max_fanout, power, noise, and seed. The default fixing data types are setup, hold, max_transition, and max_capacitance.

`-leakage_scenario` *scenario_name*

Specifies a leakage scenario name to generate leakage power data.

`-dynamic_scenario` *`scenario_name`*

Specifies a dynamic scenario name to generate dynmaic data.

`-pba_mode none | path | exhaustive | ml_exhaustive`

Specifies one of the following timing analysis modes:

- *none* (the default) - Disables path-based analysis and enables ordinary graph-based analysis. This is the fastest mode.
- *path* - Performs path-based analysis on paths after they have been gathered by graph-based analysis, producing more accurate timing results for those paths.
- *exhaustive* - Performs an exhaustive path-based analysis to determine the truly worst-case paths in the design. This is the most accurate and most computation-intensive mode.
- *ml_exhaustive* - Performs machine learning based on exhaustive path-based analysis. This mechanism deploys machine learning techniques to trade runtime versus accuracy during the design flow. Accuracy and runtime will match the *-pba_mode exhaustive* option as the design approaches signoff.

`-pt_version specific | compatible`

Specifies whether to save a version-specific or version-compatible session

- *specific* Saves the session in a form that can be restored into the same PrimeTime release. Although this is the default, this keyword provides an explicit way to specify the default. A timing update is not needed when restoring the session.
- *compatible* Saves the session in a form that can be restored into the same or later PrimeTime releases. A timing update is always needed when restoring the session.

`-link_session` *`dir_name`*

Specifies a saved session directory name to create a symbolic link. With this option specified, the *write_eco_session* command creates a symbolic link to an existing PrimeTime session instead of creating a new session. This option is useful to save disk space. Note that this option only works in a scalar session.

`-spef_data` *`list_of_spef_files`*

Specifies a list of RC data files (SPEF or GPD) to replace the *write_eco_session* command corresponding with the RC data. This option is useful to resolve missing original RC data and for HyperScale design. Note that this option only works in a scalar session.

`-netlist_data` *`list_of_netlist_files`*

Specifies a netlist file list to replace the *write_eco_session* command with the corresponding netlist data. This option is useful for a HyperScale design. Note that this option only works in a scalar session.

`directory` *`dir_name`*

Specifies a directory path name to write a session.

Description

Use this command to create a repository of data for the PrimeClosure tool to perform ECO fixing. Use the *dir_name* option to specify the name of the directory to write the session. If the directory does not exist, a new directory is created and the data for the session is written into the new directory.

By default, the command writes PrimeTime sessions for PrimeClosure Distributed Multi Scenario Static Analysis (DMSA) mode and timing data for Single Machine Static Analysis (SMSA) mode. You can specify the *-include* option to write selected session data types. These session data types are pt_session, eco_data, design, netlist, SPEF, libraries. If the *-include* option is not specified, the default data types are pt_session and eco_data.

Scalar V.S. Distributed Multi Scenario Analysis (DMSA) mode

If the command is used in a scalar session, the directory contains the session data only. If the command is used in a DMSA session, the directory contains all scenaro sessions.

In the PrimeClosure tool, use the *read_eco_session* command to read the session written by this command. If the session is written from a DMSA session, the *read_eco_session* command reads all scenarios automatically. If the session is written from a scalar session, use the *-scenario_name* option to specify the scenario name associated with the session data.

Portability

The command uses symbolic links to existing files, such as verilog and parasitics files, instead of saving them in the session to save disk space. However, if those files are moved or deleted, the PrimeClosure tool will fail due to missing data. Also, if you want to move the session from one network to another network with different UNIX paths, the PrimeClousre tool will fail.

If you specify the 'design' type with the *-include* option, the design data, such as netlist and SPEF, are written to the session so that the session can be portable to other locations without missing data. You can specify either the netlist or SPEF design data, if needed.

Saving disk space

The command uses a PrimeTime infrastructure to save PrimeTime sessions for the PrimeClosure DMSA mode. If you have many scenarios, the pt_session can take a large

portion of disk space. You can save disk space by creating a symbolic link to existing PrimeTime sessions as long as the existing session can be assessed by the PrimeClosure tool. Use the *-link_session* option to specify the existing PrimeTime session.

Note that the *-link_session* option works only in a scalar single scenario session.

HyperScale design

If PrimeTime session contains a HyperScale design, the netlist and SPEF file paths written in the mcmm_data file under the session directory will be incomplete. If you read the session in PrimeClosure, SMSA mode considers missing blocks as black boxes. Use the *-spef_data* and *-netlist_data* options to overwrite the path list.

Note that *-spef_data* and *-netlist_data* options work only in a scalar single scenario session.

Examples

If you are running a DMSA static timing analysis (STA) session, you can run the following command in a PrimeTime manager.

```
pt_shell> write_eco_session $my_dmsa_eco_session
```

Since the command was executed in a PrimeTime master, the $my_dmsa_eco_session session directory contains multi-scenario data with subdirectories named after each scenario.

If you are running STA separately for different scenarios (scalar mode), you can still generate a session using the same *write_eco_session* command. For example:

```
pt_shell> write_eco_session $my_dir/eco_session_${scenario}
```

Suppose you have two PrimeTime scenarios, setup and hold. In the setup scenario, the following command writes a session named $my_setup_eco_session:

```
pt_shell> write_eco_session $my_setup_eco_session
```

In the hold scenario, the following command writes a session in a separate directory named $my_hold_eco_session:

```
pt_shell> write_eco_session $my_hold_eco_session
```

---

write_edge_annotation

Writes out boundary leakage edge annotation information in edge annotation file.

Syntax

status *write_edge_annotation*

```
[-edge_annotation_files file_fmt]
```

Feedback

Data Types

file_fmt	string

Arguments

`-edge_annotation_files` *file_fmt*

Specifies the name of the edge annotation file.

Description

PrimePower supports boundary leakage power analysis and reporting through edge back annotation flow. The *write_edge_annotation* command writes out boundary leakage edge annotation information in edge annotation file. The edge annotation information can be read in later in the design and boundary leakage analysis can be done using the edge annotation information.

In order to write out edge annotation information, boundary leakage analysis must be run in the design.

Examples

The following example writes out boundary leakage edge annotation information in the edge annotation file

```
pt_shell> write_edge_annotation -edge_annotation_files
 { edge_annotation }
```

See Also

- set_eco_options
- check_eco
- read_context_leakage_data
- read_edge_annotation

---

write_hier_data

Writes out model data for the current design to be used for top-level HyperScale analysis, or writes out context data for HyperScale subblocks in the current design.

Syntax

status *write_hier_data*

```
[-netlist]
[-parasitics spef_format | gpd_format]
[-config]
```

```
[-exclude parasitics_data | context_data]
[-include model_reload_data | model_constraints_data]
[-script_format ptsh | sdc]
dir_name
```

Data Types

dir_name string

Arguments

`-netlist`

Writes out a netlist for the HyperScale block model interface instead of the HyperScale block model. Use this option when you want to export the HyperScale block interface netlist in Verilog format to use with other tools.

Writing out the netlist s not required in the PrimeTime HyperScale analysis flow.

`-parasitics spef_format | gpd_format`

Specifies the format, either SPEF or GPD, for the parasitic data written to the hierarchical data directory. The default is GPD. This is the parasitic data of the HyperScale block model interface.

`-config`

Writes a HyperScale configuration script instead of a HyperScale block model or block context. This script contains commands that enable HyperScale hierarchical analysis and set the hierarchical configuration (using the *set_hier_config* command). You can optionally use this script to prepare the same analysis in a future session.

`-exclude parasitics_data | context_data`

Use this option to skip writing of parasitic data for the interface nets of HyperScale block models or skip writing top-level context data for block-level analysis.

This option is typically used to save disk space when the parasitics are already available for the entire block.

When available block-level parasitics are used, it is extremely important to provide exactly the same block-level parasitics for block-level analysis that were used in top-level analysis. HyperScale top-level analysis will automatically annotate only the interface nets and ignore internal nets in the full-block parasitics data. Potential accuracy loss can occur if inconsistent block-level parasitics are used.

`-include model_reload_data | model_constraints_data`

Use the *model_reload_data* value to write out additional model reload data in tool-created directory named 'RELOAD'. You can directly load this data by using the *read_hier_data* command in a separate analysis session to check the interface data.

Use the *model_constraints_data* value to write out a constraint file that is a partial version of the full-block constraints supplied. This exports only the constraints relevant to the list of objects HyperScale considers as the block interface, either automatically by default, or as configured during block model abstraction.

The *model_reload_data* value does not affect the data in the MODEL directory; it writes out extra side data used to load the block model into a separate run with only the interface circuit in the MODEL and without providing a top-level circuit to instantiate the block model.

The *model_reload_data* option requires additional runtime and disk storage. It is recommended for debugging purposes only.

The *model_constraints_data* value must be used together with the *-netlist* option.

`-script_format ptsh | sdc`

Specifies the output format for the instance script. Allowed values are *ptsh* (the default) and *sdc* for pt_shell.

`dir_name`

Specifies the directory in which to write the block model, block model netlist, block model parasitic data, or lower-level block context. Any existing data of the same type in this directory is overwritten. For this reason, be sure to specify a different dedicated directory for each individual HyperScale block and for the top-level design.

Description

The *write_hier_data* command writes out hierarchical interface data in the discrete HyperScale analysis flow. In this flow, you use a separate analysis session for each lower-

level HyperScale block and for the top level. The data written by this command allows the exchange of interface data between the block-level and top-level analysis sessions:

- In each block-level session, the *write_hier_data* command writes out the HyperScale model for the block. In a subsequent top-level analysis session, the tool uses the HyperScale block model as directed by the *set_hier_config* command.
- In a top-level analysis session, the *write_hier_data* command writes out the context for each lower-level HyperScale block, as defined by the *set_hier_config* command. In each subsequent block-level analysis session, you read in the context data for that block by using the *read_context* command.

Block-Level and Top-Level Analysis

The current design is treated as a block-level design if it does not contain HyperScale blocks (in other words, when the *set_hier_config* command is not used in the session), or if the *read_context* command is used, or if you set the *timing_save_hier_model_data* variable to *true*. In any of these cases, the *write_hier_data* command writes out a HyperScale block model for the current design. This model can be used for analysis at the next higher HyperScale level of hierarchy.

The current design is treated as a top-level design if it contains HyperScale blocks, as specified by the *set_hier_config* command. In that case, the *write_hier_data* command writes out the context for the lower-level HyperScale blocks. The context information establishes the constraints for individual analysis of the lower-level HyperScale blocks.

A mid-level design can be both a block-level design (relative to its parent) and a top-level design (relative to its HyperScale blocks). In that case, the *write_hier_data* command writes out both a HyperScale block model for the current design and the context for the lower-level HyperScale blocks. When the *-exclude context_data* option is used, the *write_hier_data* command writes out only the model data. You can also use the *write_context* command instead.

During top-level or mid-level design analysis, there can be situations when you do not need to write out the context information for the lower-level HyperScale blocks or perform scope checking for those blocks. To save runtime, you can prevent context generation and scope checking by setting the *timing_save_hier_context_data* variable to *false*.

Hierarchical Data Directory

In each *write_hier_data* command, you must specify the directory in which to store the data. In this directory, the command writes out the HyperScale model data to a subdirectory named MODEL, or writes out the context data to a subdirectory named CONTEXT, or writes out the HyperScale model netlist and parasitic data directly. The command overwrites any existing data of the same type in the directory.

The tool automatically manages the model data stored in the model directory tree, including reading, writing, updating, and deleting of the model contents. Do not attempt to directly modify the model files or subdirectories, as this can corrupt the database.

Block Model Netlist and Parasitics

In a block-level session, you can optionally write out the netlist and parasitic data for the block model by using a separate *write_hier_data* command with the *-netlist* and *-parasitics* options. This data represents part of the block that is retained and visible in the HyperScale model for top-level analysis, roughly corresponding to the interface logic of the block. You can export this information to use with other tools; it is not needed in the HyperScale analysis flow.

The netlist and parasitic data can be written only with no subblocks loaded as HyperScale blocks, and only after a timing update. The netlist is written in Verilog format and the parasitic data is written in Galaxy Parasitic Database (GPD) or SPEF format, as specified by the *-parasitics* argument. Text files are written in compressed gzip format.

Block-Level UPF Data

In a block-level session, the *write_hier_data* command automatically extracts any UPF data for the HyperScale block model. During top-level analysis, the tool automatically loads the extracted block UPF data along with the rest of the HyperScale model data, and it ignores any attempt to load UPF data for HyperScale blocks using the *load_upf* command.

Usage Notes

In a top-down hierarchical flow, from a top-level analysis, you might want to generate the only the constraints (rather than the full context) for lower-level blocks for which there are not yet any HyperScale models. In that case, do not use the *write_hier_data* command. Instead, set the *hier_characterize_context_mode* variable to *constraints_only*, and then use the *characterize_context* and *write_context* commands to generate and write out the initial constraints for the lower-level blocks.

Examples

In a discrete hierarchical analysis flow, you analyze each HyperScale block and the top-level in a separate session. The *write_hier_data* command writes out the hierarchical data from each session, allowing the tool to exchange data between sessions at different hierarchical levels.

In the following example, a top-level design called TOP contains two HyperScale blocks, BLKA and BLKB.

BLKA block-level analysis session:

```
set_app_var hier_enable_analysis true
 ...
read_verilog BLKA.v
```

```
link_design BLKA
 ...
# Read context data from top-level analysis session, if available
read_context $topContextDir
 ...
update_timing
 ...
# Write HyperScale block model data for BLKA
write_hier_data $model_A
# Write block model interface netlist & parasitics (optional)
write_hier_data -netlist -parasitics spef_format $model_A
```

BLKB block-level analysis session:

```
set_app_var hier_enable_analysis true
 ...
read_verilog BLKB.v
link_design BLKB
 ...
# Read context data from top-level analysis session, if available
read_context $topContextDir
 ...
update_timing
 ...
# Write HyperScale block model data for BLKB
write_hier_data $model_B
# Write block model interface netlist & parasitics (optional)
write_hier_data -netlist -parasitics spef_format $model_B
```

Top-level analysis session using HyperScale BLKA/BLKB models:

```
set_app_var hier_enable_analysis true
 ...
set_hier_config -block BLKA -path $model_A
set_hier_config -block BLKB -path $model_B
 ...
read_verilog TOP.v
link_design TOP
 ...
update_timing
 ...
# Write context data for BLKA and BLKB
write_hier_data $topContextDir
```

You can run these scripts multiple times to analyze design changes or refine the constraints. The block-level and top-level analysis sessions exchange hierarchical data by using the *write_hier_data*, *read_context*, and *set_hier_config* commands.

Feedback

See Also

- read_context
- set_hier_config
- write_context
- hier_enable_analysis
- timing_save_hier_context_data
- timing_save_hier_model_data

---

write_ilm_netlist

Writes out a flattened Verilog netlist corresponding to the interface logic for the current design.

Syntax

int *write_ilm_netlist*

```
[-include_all_net_pins]
[-verbose]
file_name
```

Data Types

file_name	string

Arguments

`-include_all_net_pins`

Indicates that all pins on nonclock interface logic nets and propagated clock interface logic nets are to be included in the netlist. By default, only pins that have the *is_interface_logic_pin* attribute are written out in the Verilog netlist. Use this option if the interface logic model (ILM) is used in non-SDF flows and delay calculation is performed using the information contained in the ILM. Preserving all pins on a net maintains correct pin capacitance information for the net. This option does not affect nonpropagated clock nets; that is, nets in the clock network that have user-specified source latency.

`-verbose`

Indicates that information about the number of cells and nets in the original design and in the ILM netlist is to be written to standard output.

file_name

Specifies the name of the output file containing the flattened Verilog netlist information.

Description

Writes out a flattened Verilog netlist corresponding to the interface logic on the current design. This represents the netlist of the interface logic model (ILM).

Use the *-include_all_net_pins* option if the ILM is used in the non-SDF flows. You can use the *-verbose* option for approximating the reduction of the ILM size when compared with the original netlist.

For a discussion of ILM creation and the associated commands, see the *identify_interface_logic* man page.

Examples

The following example writes out a Verilog netlist that includes statistics comparing the ILM with the original netlist.

```
pt_shell> write_ilm_netlist -verbose model.v
```

The following example writes out a Verilog netlist while keeping all pins on interface logic nets in the netlist.

```
pt_shell> write_ilm_netlist -include_all_net_pins model.v
```

See Also

- get_ilm_objects
- identify_interface_logic
- write_ilm_parasitics
- write_ilm_script
- write_ilm_sdf

---

write_ilm_parasitics

Writes out in Standard Parasitic Exchange Format (SPEF) or Synopsys Binary Parasitic Format (SBPF) all annotated parasitics on the interface logic for the current design.

Syntax

int *write_ilm_parasitics*

```
[-input_port_nets]
[-constant_nets]
[-compress compression]
[-format SPEF | SBPF]
file_name
```

Data Types

compression	string
file_name	string

Arguments

`-input_port_nets`

Indicates that parasitic information is to be written out for nets connected to the input ports on the current design. By default, this information is not written out.

`-constant_nets`

Indicates that parasitic information is to be written out for constant nets. By default, this information is not written out.

`-compress` *compression*

Specifies that the file should be compressed. The only valid value for *compression* is *gzip*.

`-format SPEF | SBPF`

Specifies the format of the file. The valid values are *SPEF* and *SBPF*. The default is *SPEF*.

file_name

Specifies the name of the output SPEF file.

Description

This command writes out in SPEF format all annotated parasitics on the current design that are defined on its interface logic. This information is used as back-annotation information for the interface logic model (ILM).

For a discussion of ILM creation and the associated commands, see the *identify_interface_logic* man page.

Examples

The following example writes to the SPEF file model.spef the parasitics that apply to interface logic.

```
pt_shell> write_ilm_parasitics model.spef
```

The following example includes parasitic information for the input ports and writes all parasitic information to the SPEF file model.spef.

```
pt_shell> write_ilm_parasitics -input_port_nets model.spef
```

See Also

- get_ilm_objects
- identify_interface_logic
- write_ilm_netlist
- write_ilm_script
- write_ilm_sdf

---

write_ilm_script

Writes constraints, assertions and exceptions for interface logic as a script for PrimeTime or Design Compiler.

Syntax

int *write_ilm_script*

```
[-instance]
[-format ptsh | dcsch | dctcl]
[-compress compression]
[-nosplit]
file_name
```

Data Types

compression	string
file_name	string

Arguments

`-instance`

Indicates that the script written out should be appropriate for use when the interface logic is instantiated at the chip level. That is, assertions and exceptions must refer to boundary pins on a block instead of referring to ports on a design. The script does not include environment information because that is defined by the chip-level design. Do not use this option to generate a script for verifying the standalone ILM against the original design.

`-format ptsh | dcsch | dctcl`

Specifies the output format for the script. Allowed values are *ptsh* (the default) for pt_shell, *dcsh* for dc_shell, and *dctcl* for dc_shell-t.

`-compress` *`compression`*

Specifies that the script is to be compressed; by default, the script is written as standard text. The only valid value for the *compression* option is *gzip*.

`-nosplit`

The *-nosplit* option prevents line-splitting. This is most useful for doing diffs on previous scripts, or for postprocessing the script.

`file_name`

Specifies the name of the script file to write. If the *-compress* option is also specified, the extension ".gz" is added if it is not already present.

Description

Writes constraints, assertions and exceptions for interface logic as a script for pt_shell, dc_shell, or dc_shell-t.

For a discussion of ILM creation and the associated commands, see the manual page for the *identify_interface_logic* command.

Examples

The following example writes out in ptsh format the assertions and exceptions that apply to interface logic, to the file model.pt. This script would be used to verify an ILM against the original netlist.

```
pt_shell> write_ilm_script model.pt
```

The following example writes out the assertions and exceptions that can be applied to the ILM when it is instantiated at the chip level.

```
pt_shell> write_ilm_script -instance model.pt
```

See Also

- get_ilm_objects
- identify_interface_logic
- write_ilm_netlist
- write_ilm_parasitics

Feedback

- write_script
- ilm_ignore_percentage

---

write_ilm_sdf

Writes out a Version 2.1-compliant Standard Delay Format (SDF) back-annotation file for the interface logic of the current design.

Syntax

integer *write_ilm_sdf*

```
[-input_port_nets]
[-output_port_nets]
[-significant_digits digits]
[-annotated]
[-no_edge]
[-compress compression]
[-version sdf_version]
file_name
```

Data Types

digits	integer
compression	string
sdf_version	string
file_name	string

Arguments

`-input_port_nets`

Indicates that the SDF file is to include delays of nets connected to input ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available.

`-output_port_nets`

Indicates that the SDF file is to include delays of nets connected to output ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available.

`-significant_digits` *digits*

Specifies the number of digits to the right of the decimal point that are to be written in SDF delay triplets. Allowed values are 0-13; the default is 3.

`-annotated`

Indicates that the SDF is to include only timing arcs that have been annotated with the *read_sdf*, *set_annotated_delay*, or *set_annotated_check* commands.

`-no_edge`

Indicates that the generated SDF is not to include any edges (posedge or negedge) for combinational IOPATHs.

`-compress` *`compression`*

Specifies that the file should be compressed. The only valid value for *compression* is *gzip*.

`-version` *`sdf_version`*

Selects which SDF version to use. Supported SDF versions are 1.0, 2.1, and 3.0. SDF version 2.1 is the default.

`file_name`

Specifies the name of the SDF file to write. If the *-compress* option is also specified, the extension ".gz" is added if it is not already present.

Description

This command writes out in SDF format information on cell and net delays for the interface logic on the current design.

For information about ILM creation and the associated commands, see the *identify_interface_logic* man page.

Examples

The following example writes out SDF information that applies to the interface logic; only SDF data that was previously back-annotated on a design is written out.

```
pt_shell> write_ilm_sdf -annotated model.sdf
```

The following example writes out an SDF file, model.sdf, for the interface logic. All delay values are included whether or not they had been back-annotated. Information is included for the input and output ports on the design. Four significant digits to the right of the decimal point are written in SDF delay triplets.

```
pt_shell> write_ilm_sdf -input_port_nets -output_port_nets
 -significant_digits 4 model.sdf
in -0.25in
```

See Also

- get_ilm_objects
- identify_interface_logic
- write_ilm_netlist
- write_ilm_parasitics

- write_ilm_script
- write_sdf

---

write_implement_changes

Writes out ECO changes performed in the current session in DEF, Verilog, or IC Compiler II Tcl script format.

Syntax

status *write_implement_changes*

```
-format icc2tcl | verilog | def
-output file_name
```

Data Types

```
file_name     string
```

Arguments

`-format icc2tcl | verilog | def`

Specifies one of the following output formats:

- *icc2tcl* - A change list script in IC Compiler II Tcl command format that implements the netlist changes
- *verilog* - The ECO-modified design netlist in Verilog format
- *def* - The ECO-modified physical design in Design Exchange Format (DEF)

`-output file_name`

Specifies the name of the output file (Tcl script, Verilog, or DEF file).

Description

The *write_implement_changes* command writes out the ECO changes that were made to the specified ECO block. The *-format* option specifies whether to write out the changes as a Tcl script file that can be run in the IC Compiler II tool, a Verilog netlist, or DEF physical data file. You can use a Verilog netlist and DEF file to export the design to other tools.

The *write_implement_changes* command attempts to merge redundant operations to reduce and simplify the change list. For example, if a cell is resized multiple times, it is simplified to a single resize operation. If a buffer is inserted later removed, the associated commands are omitted.

Feedback

Examples

The following command writes out the change list file as an IC Compiler II Tcl script. Running the script in IC Compiler II performs all the ECO netlist changes and physical changes of the session.

```
prompt> write_implement_options -format icc2tcl -name my_eco_changes.tcl
```

The following commands write out the ECO-modified design netlist in Verilog format and the ECO-modified physical design in DEF format.

```
prompt> write_implement_changes -format verilog -name post_eco.v

prompt> write_implement_changes -format def -name post_eco.def
```

See Also

- implement_eco
- set_implement_options
- write_changes

---

write_interface_timing

Generates an interface timing ASCII report for a gate-level netlist or an extracted timing model (ETM) design.

Syntax

int *write_interface_timing*

```
file_name
[-ignore_ports port_list]
[-significant_digits digits]
[-nosplit]
[-timing_type slack | arc]
[-verbose]
[-include incl_list]
[-latch_level level]
[-sms_scenarios sms_scenarios_list]
```

Data Types

digits	int
file_name	string
incl_list	list
level	int
port_list	list
sms_scenarios_list	collection

Feedback

Arguments

file_name

Specifies the name of the file to which the interface timing information is to be written.

-ignore_ports *port_list*

Specifies a list of ports that are to be excluded from the timing file. By default, all ports are included.

-significant_digits *digits*

Specifies the number of digits to the right of the decimal point that are to be reported following the method used in the *report_timing* command. Allowed values are 0-13.

The default is the maximum of six and the value of the *report_default_significant_digits* variable. Use this option if you want to override the default. Fixed-precision floating-point arithmetic is used for delay calculation; thus, the actual precision might depend on the platform.

-nosplit

Prevents line-splitting. This is most useful for doing diffs on previous scripts or for postprocessing the script.

-timing_type slack | arc

Selects the type of timing data section to be reported.

- *slack* (default) - Reports the slack associated with interface timing relationships.
- *arc* - Reports the timing arc values for interface timing relationships.

Slack values are a function of the external environment (clocks, arrival times), but arc values are not.

If you are validating an ETM that gave a MEXT-54 warning when generated, do not use arc values. When the MEXT-54 warning is given, some of the ETM arcs have been adjusted by a multiple of the clock period and does not match the arc values of the original netlist.

-verbose

Shows details of the progress of the *write_interface_timing* command and provides data on CPU time and memory usage. For longer runs, it shows a time estimate of completion.

`-include` *`incl_list`*

Selects the type of data to analyze. You can specify the following values for the *incl_list*:

- *timing* (default) - Reports the slack/arc, capacitance, transition time, and design rules sections.

`-latch_level` *`level`*

Specifies the number of levels of latch borrowing that are to occur at the interface of a design. The default is 12.

`-sms_scenarios` *`sms_scenarios_list`*

Writes interface timing only for the specified SMVA/SMS scenarios. This collection is created by the *get_sms_scenarios* command. The default is to write interface timing for all SMS scenarios relevant to the design.

In an SMVA/SMS analysis, the *write_interface_timing* command creates a separate file for each SMS scenario, augmenting the file name with a description of the voltage configuration as follows:

```
  <name>_<supply_group1>_<ref_name1>[..._<supply_groupN>_<ref_nam
eN>]
```

where *supply_group1,.,supply_groupN* are the supply groups included in the voltage configuration, and *ref_name1,..ref_nameN* are their corresponding voltage reference names specified with the *set_voltage_levels* command.

Description

This command generates an interface timing ASCII report for a gate-level netlist, or an extracted timing model (ETM) design. The report is a file containing interface timing, slew, capacitance, design rule, and noise information for a gate-level netlist or an ETM design. The command performs an implicit *update_timing* and *update_noise*, if necessary, for the selected data sections. A return code of *1* indicates that the command ran to completion; a return code of *0* indicates an error.

The output report file contains the following sections:

Slack - Shows worst-case slacks for combinational paths from an input port to an output port, for input-to-register paths from an input port to each clock used to clock the input port, and for output-to-register paths from each clock used to clock an output port to the output port.

For each port-to-port, register-to-port, and port-to-register timing relationship, it reports slacks for the following possible arc types: *min_seq_delay*, *max_seq_delay*,

min_combo_delay, *max_combo_delay*, *setup*, *hold*, *recovery*, *removal*, *clock_gating_setup*, and *clock_gating_hold.*

Arc - Shows worst-case setup/hold/delay arc values for combinational paths from an input port to an output port, for input-to-register paths from an input port to each clock used to clock the input port, and for output-to-register paths from each clock used to clock an output port to the output port.

For each port-to-port, register-to-port, and port-to-register timing relationship, it reports arc values for the following possible arc types: *min_seq_delay*, *max_seq_delay*, *min_combo_delay*, *max_combo_delay*, *setup*, *hold*, *recovery*, *removal*, *clock_gating_setup*, and *clock_gating_hold.*

Transition Time - Shows actual transition times on all ports for the four delay types: *min_fall*, *min_rise*, *max_fall*, and *max_rise*.

Input Capacitance - Shows actual total (lumped) or effective capacitances on boundary nets connected to each port.

Design Rule - Shows design rule information for all ports: max_capacitance, min_capacitance, max_transition, max_fanout, and fanout_load.

Noise Detection - Shows noise slack information for all input ports for the four noise regions: below low, above low, below high, above high.

Noise Calculation - Shows driver steady-state resistance for all output ports for the four noise regions: below low, above low, below high, above high.

To compare two report files generated by the *write_interface_timing* command, use the *compare_interface_timing* command.

Examples

The following example writes timing information for the current design to the model.rpt file.

```
pt_shell> write_interface_timing -timing_type arc model.rpt
```

See Also

- compare_interface_timing
- update_noise
- update_timing
- report_default_significant_digits

---

write_parasitics

Writes out annotated parasitics information for the current design.

Feedback

Syntax

status *write_parasitics*

```
-format SPEF | GPD
-no_name_mapping
-nets net_list
file_name
```

Data Types

net_list	list
file_name	string

Arguments

`-format SPEF | GPD`

Specifies the format of the output parasitics file. Currently, the only allowed values are Standard Parasitic Exchange Format (*SPEF*) and Galaxy Parasitics Database (*GPD*).

`-no_name_mapping`

This option is only for SPEF. When specified for SPEF, it means that name maps should not be used for generating SPEF file. By default, a name map section is always generated.

`-nets net_list`

Specifies the list of nets to output parasitics in the parasitics file. This option is supported for both SPEF and GPD.

file_name

Specifies the name of the output parasitics file.

Description

The *write_parasitics* command writes all parasitics annotated on the current design to the file specified in the *file_name* argument. This command supports the SPEF and GPD formats, and a *PrimeTime SI* license is required for GPD.

Binary parasitics are useful for analysis iterations. You can read parasitics in one format (for example, SPEF), then use the *write_parasitics* command to output the parasitics in GPD format, which loads much faster than ASCII formats and is much more compact. The *write_parasitics* command can write detailed RC, PI model, and lumped-capacitance networks.

SPEF writing can be useful for debugging if you read parasitics into PrimeTime through GPD.

Feedback

Examples

The following example reads the top.spef parasitics file and then writes it out in GPD format:

```
pt_shell> read_parasitics top.spef
****************************************
Report : read_parasitics /u/designs/top.spef
Design : TOP
****************************************
        0 error(s)
        Format is SPEF
        Number of annotated nets          :        66
        Number of annotated capacitances  :      1390
        Number of annotated resistances   :      1333
        Number of annotated RC pi models  :         0
        Number of annotated Elmore delays :         0
1
pt_shell> write_parasitics -format GPD top.gpd
1
pt_shell> write_parasitics -format SPEF top.spef
1
```

See Also

- read_parasitics

---

write_physical_annotations

Writes annotated delays and parasitics for a hierarchical cell.

Syntax

string *write_physical_annotations*

```
[-sdf sdf_file]
[-version sdf_version]
[-nets_only]
[-cells_only]
[-boundary_nets]
[-parasitics parasitics_file]
[-append script_file]
[-format ptsh | dcsh | dctcl]
cell
```

Data Types

```
sdf_file              string
sdf_version           string
parasitics_file       string
```

```
script_file             string
cell                    string
```

Arguments

`-sdf` *sdf_file*

Indicates that annotated delays are to be written in Standard Delay Format (SDF) to the specified *sdf_file*. You must specify either *-sdf* or *-parasitics*, but you cannot specify both. If you specify either the *-version*, *-nets_only*, or *-cells_only* option, you must also specify the *-sdf* option.

`-version` *sdf_version*

Specifies the version of the SDF to use. Allowed values are *1.0* and *2.1* (the default). If you specify the *-version* option, you must also specify the *-sdf* option.

`-nets_only`

Indicates that only delays annotated on nets be written out as SDF. By default, all annotated delays are written out. If you specify the *-nets_only* option, you must also specify the *-sdf* option.

`-cells_only`

Indicates that only delays and timing checks annotated on cells be written out as SDF. By default, all annotated delays are written out. If you specify the *-cells_only* option, you must also specify the *-sdf* option.

`-boundary_nets`

Indicates that parasitics annotated on boundary nets of the specified cell must also be written out to the parasitics file. By default, only the annotated parasitics of internal nets are written out. Do not use this option if you issue this command in conjunction with the *characterize_context* command to avoid overwriting characterized capacitance on boundary nets. If you specify the *-boundary_nets* option, you must also specify the *-parasitics* option.

`-parasitics` *parasitics_file*

Indicates that annotated capacitance and resistance on nets are to be written to the specified *parasitics_file* as a script. Annotated capacitance is written out as a *set_load* command for for pt_shell and dc_shell. Annotated resistance is written out as a *set_resistance* command. You must specify either the *-sdf* or *-parasitics* option, but you cannot specify both options. If you specify the *-boundary_nets* option, you must also specify the *-parasitics* option.

`-append` *script_file*

Appends commands to import the generated files in the given script file. The SDF file is read using the *read_sdf* command in pt_shell and the *read_timing* command in dc_shell. The parasitics file is included in the script file.

`-format ptsh | dcsh | dctcl`

Specifies the format in which to output the commands in the script file and the parasitics file. Allowed values are *ptsh* (the default), *dcsh*, or *dctcl*.

cell

Specifies the name of the cell for which to export the annotations. The cell must be hierarchical.

Description

The *write_physical_annotations* command writes physical information for a hierarchical block in the design. The exported information includes annotated net and cell delays using the Standard Delay Format (SDF). Annotated parasitics can also be exported as a series of pt_shell or dc_shell commands.

The *write_physical_annotations* command is most commonly used in conjunction with the *characterize_context* and *write_context* commands to export timing and physical information for a block from the chip-level environment to module level tools. A typical sequence of commands is to first characterize the timing environment of a block using the *characterize_context* command and write this information as a dc_shell script using the *write_context* command. The *write_physical_annotations* command is then used to export annotated delays and parasitics for internal nets of the block.

To avoid overwriting characterized capacitance on boundary nets, do not specify the *-boundary_nets* option. You can use the *-append* option to append commands to read the SDF and include the parasitics script at the end of the script file generated by the *write_context* command.

Examples

The following command writes the annotated delays on nets for the hierarchical cell 'I1':

```
pt_shell> write_physical_annotations \\
            -sdf I1.sdf -nets_only I1
```

The following command is used in conjunction with the *characterize_context* and *write_context* commands to export the timing and physical environment for cell 'I2' as a dc_shell script. The command uses the *append* option to augment the script generated by the write_context command with commands to import the generated delay and parasitic files.

```
pt_shell> characterize_context I2

pt_shell> write_context -format dcsh \\
          -out I2.dcsh I2

pt_shell> write_physical_annotations \\
          -sdf I2.sdf -parasitics I2.rc \\
```

```
-format dcsh -append I2.dcsh \\
I2
```

See Also

- characterize_context
- read_sdf
- remove_context
- report_context
- set_load
- set_resistance
- write_context
- write_sdf

---

write_rh_file

Writes timing input file for ANSYS RedHawk solution.

Syntax

string *write_rh_file*

```
[-filetype irdrop | jitter ]
[-significant_digits digits]
-output file_name
```

Data Types

digits	int
file_name	string

Arguments

`-significant_digits` *digits*

Specifies the number of digits to the right of the decimal point that are to be written in the file. The default is the value of variable *report_default_significant_digits*.

`-filetype irdrop | jitter`

The filetype to be written out. If not specified, the irdrop file will be written.

file_name

Specifies the name of the timing file to be written for ANSYS RedHawk

Description

This command writes the timing input file for ANSYS RedHawk to a compressed disk file with ".gz" suffix.

Examples

The following example writes timing information for the current design of current scenario to a disk file called test_rh.irdrop.gz

```
prompt> write_rh_file -output test_rh.irdrop
```

---

write_rh_power_file

Writes an instance power file for the ANSYS RedHawk solution.

Syntax

string *write_rh_power_file*

```
[-output file_name]
[-extended_rh_file]
[-exclude cell_list]
[-compress]
[-exclude_pg_type pg_type_list]
[-include_ground_current]
```

Data Types

```
file_name     string
cell_list list
pg_type_list list
```

Arguments

`-output file_name`

Specifies the name of the instance power file to be written for ANSYS RedHawk solution.

`-extended_rh_file`

Supports the more comprehensive 9-column format.

`-exclude cell_list`

Excludes cells (memory or macros, etc) from being in the instance power file.

`-compress`

Writes the instance power file in the compressed (.gz) format.

`-exclude_pg_type`

Excludes PG pin types (bias, internal_power, or both) from being including the instance power file. Valid values are *bias* and *internal_power*.

`-include_ground_current`

Includes the ground current information in the instance power file.

Description

This command writes the instance power file for the ANSYS RedHawk solution.

Examples

The following example writes power information for the current design to a disk file called test_rh.ipf, excluding a list of cells using the -exclude option.

```
pt_shell> write_rh_power_file -output test_rh.ipf -exclude [get_cell *]
```

The following examples write the instance power file in the compressed format.

```
pt_shell> write_rh_power_file -output test_rh.ipf -compress

pt_shell> write_rh_power_file -output test_rh.ipf.gz -compress
```

The following examples write the instance power file excluding PG pin types of bias, internal_power or both.

```
pt_shell> write_rh_power_file -output test_rh.ipf -exclude_pg_type {bias}

pt_shell> write_rh_power_file -output test_rh.ipf -exclude_pg_type
 {internal_power}

pt_shell> write_rh_power_file -output test_rh.ipf -exclude_pg_type {bias
 internal_power}
```

The following example runs the *write_rh_power_file* command with the -include_ground_current and -extended_rh_file options and generates an instance power file which lists the instance power and current values by different power and current types per power and ground pins.

```
pt_shell> write_rh_power_file -output test -include_ground_current
 -extended_rh_file

Instance Pin   Power/  Toggle Fre-   Total     Switching Internal Leakage
Name     Name  Ground  Rate   quency Power/    Power/    Power/   Power/
               Voltage               Current   Current   Current
 Current
-------------------------------------------------------------------------
u0       VDD   0.9     0.1    0      1.844e-07 8.236e-08 1.018e-07
 2.13e-10
```

```
u0       VSS   0       0.1    0      2.134e-07 9.532e-08 1.179e-07
 2.465e-10
u1/u1    gnd   0       0.1    0      9.172e-07 8.269e-08 5.404e-07
 2.942e-07
u1/u1    vddi  1       0.1    0      1.15e-07  0         0
 1.15e-07
u1/u1    vddo  1       0.1    0      9.627e-07 9.923e-08 6.485e-07
 2.15e-07
u1/u2/u2 VDD   1       0.1    0      1.515e-05 1.504e-05 1.141e-07
 2.13e-10
u1/u2/u2 VSS   0       0.1    0      1.754e-05 1.74e-05  1.32e-07
 2.465e-10
```

See Also

- update_power

---

write_saif

Writes a backward Switching Activity Interchange Format (SAIF) file.

Syntax

int *write_saif*

```
file_name
[-cells cell_list]
[-derate_glitch value]
[-rtl]
[-propagated]
[-exclude_sdpd]
[-no_hierarchy]
[-compress compression]
[-conflict]
[-event_based]
[-include_sequential_inputs]
[-exclude_hier_pins]
[-keep_input_saif_sdpd_info]
```

Data Types

file_name	string
cell_list	list
value	float

Arguments

file_name

Specifies the name of the output SAIF file.

`-cells` *`cell_list`*

Specifies an optional list of hierarchical instances for which the SAIF file is generated. If this argument is not specified, the SAIF file is generated for the current instance.

`-derate_glitch` *`value`*

Specifies a derating factor value to be used for converting part of glitches into inertial glitches. If this argument is not specified, a default derating factor of 0.5 is used.

`-rtl`

Specifies that the generated SAIF file contains the switching activity for the synthesis invariant objects. These are the design ports, hierarchical cell pins, and outputs of sequential and tri-state cells. Note that you should not use the *-rtl_direct* option of the *read_saif* command when reading SAIF files generated by the *write_saif* command.

`-propagated`

Propagates the user-annotated switching activity to estimate the switching activity of unannotated objects, and generates a SAIF file containing both the annotated and estimated switching activity. When this option is not specified, the *write_saif* command generates a SAIF file containing only the user-annotated switching activity.

`-exclude_sdpd`

Inhibits the state-dependent and path-dependent (SDPD) switching activity generation. If you do not specify this option, the *write_saif* command outputs SDPD information by default. When the *write_saif* command is called with the *-propagated* option, the estimated SDPD information is included in the SAIF file.

`-no_hierarchy`

Specifies that the SAIF file contains switching activity information for only the top-level design objects. When this option is not specified, the *write_saif* command generates a SAIF file containing switching activity information for all the objects in the hierarchy.

`-compress`

Specify that the output saif file should be compressed. The only valid value for compression is gzip.

`-conflict`

Specify that the output saif file contains conflict switching activities only. This option is valid under merged fsdb/saif flow which is enabled when

power_enable_merged_fsdb variable is *true*. Also user should specify conflict instances using *set_power_analysis_options -conflict_activity_cells*.

`-event_based`

Writes event based saif file from RTL fsdb. update_power has to be done before using this option. *-propagated* is mandatory to be mentioned along with this option.

`-include_sequential_inputs`

This option enables writing toggle information of input pins for sequential devices. update_power has to be done before using this option. *-rtl* is mandatory to be mentioned along with this option.

`-exclude_hier_pins`

This option enables skipping writing of toggle information of hierarchical pins. update_power has to be done before using this option. *-rtl* is mandatory to be mentioned along with this option.

`-keep_input_saif_sdpd_info`

This option enables retaining the sdpd information of sequential and memory cells from the input saif read when used with '-rtl' option. *-rtl* is mandatory to be mentioned along with this option.

Description

The *write_saif* command generates a backward SAIF file containing the user-annotated and propagated simple and SDPD switching activity in the current design.

The top-level instance name in the generated SAIF file is the current instance name. You must specify this name as the instance name when reading the SAIF file with the *read_saif* command.

By default, the saif file is written as simple text. Optionally, the file can be compressed using the -compress option.In some cases, a file extension is appended to file_name if it is omitted. For example, with gzip formatted files, if file_name does not end with ".gz", then ".gz" is appended to the file name.

Examples

The following example shows how a SAIF file containing the user-annotated switching activity information on all design objects:

```
pt_shell> write_saif file
```

The following example shows how a SAIF file containing both user- annotated and propagated non-SDPD switching activity information can be generated:

```
pt_shell> write_saif file -propagated -exclude_sdpd
```

Feedback

The following example shows how a SAIF file can be compressed.

```
pt_shell> write_saif file -compress gzip
```

The following example shows how an event based SAIF file can be written from RTL fsdb.

```
pt_shell> write_saif file -propagated -event_based
```

The following example shows how to enable additional writing of toggle information of input pins of sequential devices to saif.

```
pt_shell> write_saif file -rtl -include_sequential_inputs
```

```
pt_shell> write_saif file -rtl -exclude_hier_pins
```

The following example shows how to retain sdpd information of memory/sequential cells from the read input saif.

```
pt_shell> write_saif file -rtl -keep_input_saif_sdpd_info
```

The following example shows how a SAIF file on a synthesized design object can be generated by the *write_saif* command and read back using the *read_saif* command:

```
pt_shell> current_design top
pt_shell> current_instance U1
pt_shell> write_saif design.saif
pt_shell> reset_switching_activity
pt_shell> read_saif design.saif
```

See Also

- get_switching_activity
- read_saif
- report_power
- report_switching_activity
- reset_switching_activity
- set_rtl_to_gate_name
- set_switching_activity
- power_enable_merged_fsdb

---

write_script

Writes design constraints as a script of commands for PrimeTime or Design Compiler.

Feedback

Syntax

int *write_script*

```
[-no_annotated_delay]
[-no_annotated_check]
[-format ptsh | dctcl]
[-output file_name]
[-compress gzip]
[-include category_list]
[-exclude category_list]
[-nosplit]
[-print_command_types]
```

Data Types

category_list	list
compression	string
file_name	string

Arguments

`-no_annotated_delay`

Specifies that *set_annotated_delay* commands are not to be written, which means that delay annotations are ignored.

`-no_annotated_check`

Specifies that *set_annotated_check* commands are not to be written, which means that check annotations are ignored.

`-format ptsh | dctcl`

Specifies the output format for the script. Valid values are *ptsh* (the default) for pt_shell and *dctcl* for dc_shell.

`-output file_name`

Specifies the name of a file to which output is to be written. The default is the standard output. If the *-compress* option is also specified, the extension ".gz" is added if it is not already present.

`-compress gzip`

Specifies that the script is to be compressed. By default, the script is written as standard text. The only valid value is *gzip*. If you specify this option, you must also specify the *-output* option.

`-include category_list`

Writes specified command categories only. The only valid values for *category_list* are *exceptions*, *derate*, *drv*, *case_analysis*, *max_time_borrow*, *clock*, *clock_gating_check*, *clock_transition*, *min_pulse_width*,*clock_latency*,

input_delay, *output_delay*, *edrc*, *cell_mode*,*disable_timing*, *clock_group*, *clock_uncertainty*, *clock_sense*, *load*, *path_group*, *drive_info*, *net_resistance*, *annotation*, *ideal_network*, *data_check*, *pvt*, *pvt_dynamic*, *false_path*,*min_delay*, *max_delay*, *multicycle_path*, *latch_loop_breaker*, *size_only*, *target_library_subset*, *clock_jitter*, *aocvm_coefficient*, *aocvm_coefficient_lib_cell*.

The *drv* category includes set_input_transition, set_driving_cell, set_load, set_max_transition, and set_max_capacitance commands.

The *exceptions* category writes only exceptions to the file.

The *cell_mode* category writes only set_cell_mode command.

The *derate* category writes only set_timing_derate command.

The *path_group* category writes only group_path command.

The *max_time_borrow* category writes only set_max_time_borrow command.

The *data_check* category writes only set_data_check command.

The *load* category writes only set_load command.

The *net_resistance* category writes only set_resistance command.

The *clock_transition* category writes only set_clock_transition command.

The *clock_gating_check* category writes only set_clock_gating_check command.

The *case_analysis* category writes only set_case_analysis command.

The *disable_timing* category writes only set_disable_timing command.

The *clock_group* category writes only set_clock_groups command.

The *clock_uncertainty* category writes only set_clock_uncertainity command.

The *clock_sense* category writes only set_sense command.

The *min_pulse_width* category writes only set_min_pulse_width command.

The *input_delay* category writes only set_input_delay command.

The *output_delay* category writes only set_output_delay command.

The *ideal_network* category writes only set_ideal_network command.

The *pvt* category writes only set_voltage command.

The *pvt_dynamic* writes only set_operating_conditions command.

The *drive_info* writes set_drive and set_driving_cell command.

The *clock* writes create_clock and create_generated_clock command.

The *clock_latency* writes set_clock_latency and set_propagated_clock command.

The *edrc* writes set_max_capacitance, set_min_capacitance, and set_max_transition command.

The *annotation* writes set_ideal_latency, set_ideal_transition, set_annotated_transition, and set_annotated_delay command.

The *false_path* writes set_false_path command.

The *min_delay* writes set_min_delay command.

The *max_delay* writes set_max_delay command.

The *multicycle_path* writes set_multicycle_path command.

The *latch_loop_breaker* writes set_latch_loop_breaker command.

The *size_only* writes set_size_only command.

The *target_library_subset* writes set_target_library_subset commad.

The *clock_jitter* writes set_clock_jitter command.

The *aocvm_coefficient* writes set_aocvm_coefficient command.

The *aocvm_coefficient_lib_cell* writes set_aocvm_coefficient command.

`-exclude` *`category_list`*

Excludes specified command categories. The only valid values for *category_list* are *exceptions*, *derate*, *case_analysis*, *max_time_borrow*, *clock*, *clock_gating_check*, *clock_transition*, *min_pulse_width*,*clock_latency*, *input_delay*, *output_delay*, *edrc*, *cell_mode*,*disable_timing*, *clock_group*, *clock_uncertainty*, *clock_sense*, *load*, *path_group*, *drive_info*, *net_resistance*, *annotation*, *ideal_network*, *data_check*, *pvt*, *pvt_dynamic*, *false_path*,*min_delay*, *max_delay*, *multicycle_path*, *latch_loop_breaker*, *size_only*, *target_library_subset*, *clock_jitter*, *aocvm_coefficient*, *aocvm_coefficient_lib_cell*.

The *exceptions* category excludes only exceptions to the file.

The *cell_mode* category excludes only set_cell_mode command.

The *derate* category excludes only set_timing_derate command.

The *path_group* category excludes only group_path command.

Feedback

The *max_time_borrow* category excludes only set_max_time_borrow command.

The *data_check* category excludes only set_data_check command.

The *load* category excludes only set_load command.

The *net_resistance* category excludes only set_resistance command.

The *clock_transition* category excludes only set_clock_transition command.

The *clock_gating_check* category excludes only set_clock_gating_check command.

The *case_analysis* category excludes only set_case_analysis command.

The *disable_timing* category excludes only set_disable_timing command.

The *clock_group* category excludes only set_clock_groups command.

The *clock_uncertainty* category excludes only set_clock_uncertainity command.

The *clock_sense* category excludes only set_sense command.

The *min_pulse_width* category excludes only set_min_pulse_width command.

The *input_delay* category excludes only set_input_delay command.

The *output_delay* category excludes only set_output_delay command.

The *ideal_network* category excludes only set_ideal_network command.

The *pvt* category excludes only set_voltage command.

The *pvt_dynamic* excludes only set_operating_conditions command.

The *drive_info* excludes set_drive and set_driving_cell command.

The *clock* excludes create_clock and create_generated_clock command.

The *clock_latency* excludes set_clock_latency and set_propagated_clock command.

The *edrc* excludes set_max_capacitance, set_min_capacitance, and set_max_transition command.

The *annotation* excludes set_ideal_latency, set_ideal_transition, set_annotated_transition, and set_annotated_delay command.

The *false_path* excludes set_false_path command.

The *min_delay* excludes set_min_delay command.

The *max_delay* excludes set_max_delay command.

The *multicycle_path* excludes set_multicycle_path command.

The *latch_loop_breaker* excludes set_latch_loop_breaker command.

The *size_only* excludes set_size_only command.

The *target_library_subset* excludes set_target_library_subset commad.

The *clock_jitter* excludes set_clock_jitter command.

The *aocvm_coefficient* excludes set_aocvm_coefficient command.

The *aocvm_coefficient_lib_cell* excludes set_aocvm_coefficient command.

`-nosplit`

Prevents line splitting. This is used for performing diffs on previous scripts or for postprocessing the script.

`-print_command_types`

Prints the option category present in the design along and the associated commands along with their respective counts. This is used for performing the write_script -include and -exclude operations. This is employed to quickly list out the option category present in the design, associated commands, and their respective count. The write_script file is extensive, user can extract specific commands of interest from the table.

While executing write_script -print_command_types, the information is automatically written out to the command.txt file. Executing write_script -print_command_types -output script.log, results in the information being written to the command_script.log file. script.log have the information related to the write_script.

Description

This command writes design constraints as a script of commands for PrimeTime or Design Compiler. The script is used to re-create design intent.

The *write_script* command writes the following information:

- Clock information: clock creation, generated clock creation, clock latency, clock uncertainty, interclock uncertainty.
- Timing information: disable timing, max time borrow. Disable timing set on library objects is written by default, but can be suppressed by setting the *write_script_include_library_constraints* variable to *false*.
- Point-to-point exceptions: false paths, min delay, max delay, multicycle paths, group path, input delay, output delay, annotated delay, and annotated checks.

Feedback

- Net attributes: capacitance, resistance. These are no longer written by default. For more information, see the man page for the *write_script_output_lumped_net_annotation* variable.
- Port attributes: fanout, capacitance, and resistance.
- Design environment: wire load model, operating condition, drive, driving cell, and input transition.
- Design rules: minimum capacitance, maximum capacitance, minimum transition, maximum transition, minimum fanout, and maximum fanout.

Examples

The following example writes the script in ptsh format to the des.pt file:

```
pt_shell> write_script -output des.pt
```

See Also

- characterize_context
- write_context
- write_script_output_lumped_net_annotation

---

write_sdc

Writes out a script in Synopsys Design Constraints (SDC) format.

Syntax

int *write_sdc*

```
file_name
[-version sdc_version]
[-compress compression]
[-include categories_list]
[-exclude category_list]
[-nosplit]
```

Data Types

```
file_name           string
sdc_version         string
compression         string
categories_list     list
```

Feedback

Arguments

file_name

Specifies the name of the file to which the SDC script is to be written.

-version *sdc_version*

Specifies the version of SDC to write. Allowed values are: *1.2*, *1.3*, *1.4*, *1.5*, *1.6*, *1.7*, *1.8*, *1.9*, *2.0*, and *latest* (the default).

-compress *compression*

Specifies that the script must be compressed. The only valid value for *compression* is *gzip*.

-include *categories_list*

Write specified command categories only. The only valid values for *categories_list* are *exceptions*, *sourcefile_refs*,*derate*,*case_analysis*, *max_time_borrow*, *clock*, *clock_gating_check*, *clock_transition*, *min_pulse_width*,*clock_latency*, *input_delay*, *output_delay*, *edrc*,*disable_timing*, *clock_group*, *clock_uncertainty*, *clock_sense*, *load*, *path_group*, *drive_info*, *net_resistance*, *annotation*, *ideal_network*, *data_check*, *pvt*, *pvt_dynamic*.

The *exceptions* category writes only exceptions to the file.

The *sourcefile_regs* category includes information about the original script location of the constraint or exception. This requires that the *sdc_save_source_file_information* variable was set to *true* when the original constraints were read. For details, see the man page.

The *derate* category writes only set_timing_derate command.

The *path_group* category writes only group_path command.

The *max_time_borrow* category writes only set_max_time_borrow command.

The *data_check* category writes only set_data_check command.

The *load* category writes only set_load command.

The *net_resistance* category writes only set_resistance command.

The *clock_transition* category writes only set_clock_transition command.

The *clock_gating_check* category writes only set_clock_gating_check command.

The *case_analysis* category writes only set_case_analysis command.

The *disable_timing* category writes only set_disable_timing command.

The *clock_group* category writes only set_clock_groups command.

The *clock_uncertainty* category writes only set_clock_uncertainity command.

The *clock_sense* category writes only set_sense command.

The *min_pulse_width* category writes only set_min_pulse_width command.

The *input_delay* category writes only set_input_delay command.

The *output_delay* category writes only set_output_delay command.

The *ideal_network* category writes only set_ideal_network command.

The *pvt* category writes only set_voltage command.

The *pvt_dynamic* writes only set_operating_conditions command.

The *drive_info* writes set_drive and set_driving_cell command.

The *clock* writes create_clock and create_generated_clock command.

The *clock_latency* writes set_clock_latency and set_propagated_clock command.

The *edrc* writes set_max_capacitance, set_min_capacitance, and set_max_transition command.

The *annotation* writes set_ideal_latency, set_ideal_transition, set_annotated_transition, and set_annotated_delay command.

`-exclude` *`category_list`*

Excludes specified command categories.

`-nosplit`

The *-nosplit* option prevents line-splitting. This is most useful for doing a diff against previous scripts or for postprocessing the script.

Description

The *write_sdc* command writes out a script file in the latest Synopsys Design Constraints (SDC) format. This script contains commands that can be used with PrimeTime or with Design Compiler. SDC is also licensed by external vendors through the Tap-in program. SDC formatted script files are read into Synopsys tools using the *read_sdc* command.

By default, the script is written as simple text. Optionally, the script can be compressed using the *-compress* option. In some cases, a file extension is appended to *file_name* if it is omitted. For example, with gzip formatted files, if *file_name* does not end with ".gz", then ".gz" is appended to the file name.

By default, the latest version of SDC is written. Some earlier versions of SDC can be written using the *-version* option.

SDC is a subset of the commands already supported by PrimeTime and the synthesis tools. Of the commands supported in the latest SDC version, the following can be written by *write_sdc*. Those added for versions 1.3 and later are noted.

General Purpose Commands:

```
list
```

Object Access Functions:

```
current_design
current_instance
get_cells
get_clocks
get_libs
get_lib_cells
get_lib_pins
get_nets
get_pins
get_ports
set_hierarchy_separator
```

Basic Timing Assertions:

```
create_clock
create_generated_clock      (1.3)
set_clock_gating_check
set_clock_latency
set_clock_transition
set_clock_uncertainty
set_false_path
set_input_delay
set_max_delay
set_min_delay
set_multicycle_path
set_output_delay
set_propagated_clock
set_min_pulse_width         (2.0)
```

Secondary Assertions:

```
set_disable_timing
set_max_time_borrow
set_data_check              (1.4)
set_timing_derate           (1.5)
```

Environment Assertions:

```
set_case_analysis
set_drive
set_driving_cell
set_fanout_load
set_input_transition
```

```
set_load
set_max_area
set_max_capacitance
set_max_fanout
set_max_transition
set_min_capacitance
set_min_fanout
set_operating_conditions
set_port_fanout_number
set_resistance
set_wire_load_min_block_size
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group
```

Similar to the *write_script* command, the *write_sdc* command writes out commands relative to the top of the design, regardless of the current instance. SDC files written by the *write_sdc* command must be read in from the top of the design.

For a complete guide to using SDC with Synopsys applications, see the *Using the Synopsys Design Constraints Format Application Note*, which is available at http://solvnet.synopsys.com.

The usage of some of these commands is restricted when reading SDC. In some cases, some options are not allowed. The *write_sdc* command supports the restricted usage by restricting what is written. The following restriction apply for SDC Version 1.3 or later:

Note: For the *set_port_fanout_number* command, the *-min* and *-max* options are not supported.

When hierarchy has been partially flattened, embedded hierarchy separators can make names ambiguous - it is unclear which hierarchy separator characters are part of the name and which are real separators. Beginning with SDC version 1.2, hierarchical names can be made unambiguous using the *set_hierarchy_separator* SDC command or the *-hsc* option available from the *get_cells*, *get_lib_cells*, *get_lib_pins*, *get_nets*, and *get_pins* SDC object access commands.

By default, PrimeTime and the synthesis tools write an SDC file using these features to create unambiguous names. It is recommended that you write SDC files that contain unambiguous names. However, if you are using a third-party application that does not fully support SDC 1.2 or later (that is, it does not support the unambiguous hierarchical names features of SDC), you can suppress these features by setting the *sdc_write_unambiguous_names* variable to *true*.

The following features are added for SDC Version 1.5 or later:

- *-clock* option of the *set_clock_latency* command
- *set_timing_derate* command with all options

Feedback

- *-clock_path -data_path -rise -fall* options of the *set_max_transition* command
- *-rise_from*, *-rise_to*, *-fall_from*, *-fall_to* options, of the *set_clock_uncertainty* command
- *-object_list* option of the *set_operating_conditions* command
- Synonyms for all of the *get_object* commands
- *get_objects -nocase* support independently with the *-regexp* option
- *get_objects -regexp* support
- *get_objects -of_objects* support for *get_cells/get_nets/get_pins*

The following features are added for SDC Version 2.0 or later:

- *-reference_pin* option of the *set_input_delay* and *set_output_delay* commands
- The *set_min_pulse_width* command

Examples

The following command writes the SDC script to the file top.sdc.

```
prompt> write_sdc top.sdc
```

See Also

- read_sdc
- write_script
- sdc_write_unambiguous_names
- sdc_save_source_file_information

---

write_sdf

Writes a Standard Delay Format (SDF) back-annotation file.

Syntax

status *write_sdf*

```
[-version sdf_version]
[-no_cell_delays]
[-no_timing_checks]
[-no_net_delays]
[-input_port_nets]
[-output_port_nets]
[-significant_digits digits]
[-enabled_arcs_only]
[-no_internal_pins]
```

```
[-instance inst_name]
[-context none | verilog | vhdl]
[-map sdf_map_file_list]
[-annotated]
[-levels level]
[-no_edge]
[-compress compression]
[-include include_list]
[-exclude exclude_list]
[-mask_violations violation_type]
[-no_negative_values values_list]
[-no_edge_merging arc_type_list]
[-delay_calculation_only_mode]
[-exclude_cells cell_list]
[-no_derates]
[-objects]
[-all_scenarios]
[-sms_scenarios sms_scenarios_list]
file_name
```

Data Types

arc_type_list	list
cell_list	list
compression	string
digits	int
exclude_list	list
file_name	string
include_list	list
inst_name	string
level	int
sdf_map_file_list	list
sdf_version	string
sms_scenarios_list	collection
values_list	list

Arguments

`-version` *sdf_version*

Selects the SDF version to use. Supported SDF versions are *2.1* (default) and *3.0*.

`-no_cell_delays`

Indicates that no cell delays are be written in the SDF file. By default, all cell pin-to-pin delays are written to the SDF file. Cell delays include the load delay of the cell. Following the SDF conventions, only cell input pin to cell output pin delays are written. For unbuffered cell output pins, delays are usually represented in libraries by a delay from an output pin to another output pin. Because this is not allowed by the SDF convention, the SDF delay for an unbuffered output is specified from cell inputs.

`-no_timing_checks`

Indicates that no cell timing checks are to be written in the SDF file. By default, all cell timing checks (for example, setup, hold, recovery, and removal) are written to the SDF file.

`-no_net_delays`

Indicates that no net delays are to be written in the SDF file. By default, all net pin-to-pin delays are written to the SDF file. If the *-input_port_nets* or *-output_port_nets* options are also specified, then only the net delays connected to the corresponding ports will be printed.

`-input_port_nets`

Indicates that the SDF file is to include delays of nets connected to input ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available. If the *-instance* option is specified, all net delays across the instance boundary leading to a pin inside the instance are included instead. If the *-levels level* option is specified, the pin must be found on any levels 1 to *level* of hierarchy.

`-output_port_nets`

Indicates that the SDF file is to include delays of nets connected to output ports of the current design. By default, these delays are not written to the SDF file because the external connectivity information for ports is not available. If the *-instance* option is specified, then all net delays across the instance boundary leading from a pin inside the instance are included instead. If the *-levels* option is specified, the pin must be found on any of levels 1 to *level* of hierarchy.

`-significant_digits` *`digits`*

Specifies the number of digits to the right of the decimal point that are to be written in SDF delay triplets. Allowed values are 0-13; the default is 3.

`-enabled_arcs_only`

Indicates that the SDF file is to contain only delays of enabled timing arcs, and is not to include delays of currently-disabled timing arcs. By default, delays of all timing arcs in the design are written to the SDF file, whether they are disabled or enabled.

`-no_internal_pins`

Indicates that the SDF file is not to include delay timing arcs from or to internal pins. Timing arcs to or from internal pins are expanded into delays from and to primary input and output of the given cell.

`-instance` *`inst_name`*

Specifies that the SDF file is to be written only for the instance named *inst_name*. By default, all pin names are relative to the *inst_name*. However, if boundary net delays are included (using the *-input_port_nets* or *-output_port_nets* option) all pin names are relative to the top design. Note that in general, if the *-input_port_nets* or *-output_port_nets* option is specified, boundary nets are written leaf-to-leaf and do not start or end on hierarchical pins. If boundary nets are required to start or end on hierarchical pins, see the *write_physical_annotations* command.

`-context none | verilog | vhdl`

Specifies the context for writing bus names in SDF. Valid values are *verilog*, *vhdl*, or *none* (the default). In the Verilog context, when pin names are displayed, the last two square bracket characters ("[" and "]") are not escaped. In the VHDL context, the last two parenthesis characters ("(" and ")") in a pin name are not escaped. In the default *none* context, all bus-delimiting characters are escaped with a backslash character ("\\"). In any context, bus-delimiting characters in instance names are always escaped. When used with the *-map* option, the *-context* option also affects the way names are printed in mapped SDF files. In the Verilog context, names are printed in %s[%d] format; in the VHDL context, names are printed in %s(%d) format.

Note: Names are affected only if they are mapped using the bus(name_to_be_changed) function in the mapping file.

`-map` *`sdf_map_file_list`*

Specifies a list of mapping files the SDF writer is to use when writing out the SDF file. A mapping file contains a user-specified format for printing SDF cell delays and constraints. When writing out SDF for a cell, the SDF writer takes the user-specified mapping, if present, to print out SDF for the cell. If no user-specified mapping is present for a cell, the SDF writer writes out SDF in the normal way.

`-annotated`

Indicates that the SDF is to include only timing arcs that have been annotated with the *read_sdf*, *set_annotated_delay*, or *set_annotated_check* command.

`-levels` *`level`*

Specifies the number of levels of hierarchy for which the SDF is written out. Level 1 means only the top design or *inst_name*. Value of N means all levels of hierarchy, 1 to N. By default, all levels of hierarchy are written out. Note that boundary net delays (using the *-input_port_nets* or *-output_port_nets* options) typically have some net arcs from or to pins outside the *inst_name* option. The location of such outside pins is not limited by the *-levels* option. That is, the

-levels and *-instance* options let you choose which boundary arcs are included, but do not restrict where the arcs lead outside of the *inst_name* option.

`-no_edge`

Indicates that the generated SDF is not to include any edges (posedge or negedge) for both combinational and sequential IOPATHs.

`-compress` *`compression`*

Specifies a format to be used to compress the file. The only valid value for *compression* is *gzip*. By default, files are not compressed.

`-include` *`include_list`*

Specifies a list of constructs and specifiers to include in the SDF file; you can specify one or more of the following:

- *SETUPHOLD* - Indicates that all SETUP and HOLD constructs are to be replaced by SETUPHOLD constructs. If a pair of setup and hold arcs are found between the same pin edges, timing information for the/both arc/arcs is written in a single SETUPHOLD construct. If a single setup/hold arc is found, then the arc will be written in a single SETUPHOLD construct with no timing information for the hold/setup portion. SETUPHOLD supports negative values.
- *RECREM* - Indicates that all RECOVERY and REMOVAL constructs are to be replaced by RECREM constructs. If a pair of recovery and removal arcs are found between the same pin edges, timing information for both arcs is written in a single RECREM construct. If a single recovery/removal arc is found, then the arc will be written in a single RECREM construct with no timing information for the removal/recovery portion. RECREM supports negative values and can be written only for version 3.0.
- *edge_specific_preset_clear* - Indicates that the generated SDF is to include the appropriate edges (posedge or negedge) for preset and clear arcs. The default behavior is not to include edge specifiers for preset and clear arcs.

`-exclude` *`exclude_list`*

Specifies a list of timing values of construct types to be either excluded from the SDF file in order to reduce its size, or to be replaced by another construct, as in the case of condelse. Allowed values are one or more of the following:

- *constant_nets* - Indicates that nets are to be omitted from the SDF file if they propagate a constant.
- *constant_delay_arcs* - Indicates that delay arcs are to be omitted from the SDF file if they propagate a constant, either from case analysis or logical inputs.

- *default_cell_delay_arcs* - Indicates that all default cell delay arcs are to be omitted from the SDF file if conditional delay arcs are present. If there are no conditional delay arcs, the default cell delay arcs are written to the SDF file.
- *wlm_load_delay* - Indicates that net delays and cell delays calculated using WLM are to be omitted from the SDF.
- *checkpins* - When Library Compiler finds both combinational and sequential arcs between pins, a checkpin is created so that all arcs are expanded in the db so that a single arc pinA->pinB is replaced by the combination of a positive unate arc pinA->pinAcheckpin1 with zero delay and an arc pinAcheckpin1->pinB with the same sense and values as the original arc. When this option is set the SDF is written out as if all checkpins were never created.
- *no_condelse* - Indicates that PrimeTime will not use the condelse statement to write out default iopaths. By default PrimeTime will replace default iopaths with the condelse construct. Specifying this option will result the condelse statement being replaced by a default iopath. This option should be used for generating simulator compatible SDF.
- *clock_tree_path_models* - Indicates that arcs that model clock tree paths are to be omitted from the SDF file.

`-mask_violations` *`violation_type`*

Adjusts the delays in the SDF so that when it is annotated back it will result in zero setup and/or hold time violations even if actual timing violations are present in the design. The SDF generated with this option should not be used for signoff purposes.

`-no_negative_values` *`values_list`*

Specifies a list of timing value types whose negative values are to be zeroed out when writing to the SDF file. Allowed values for timing values are timing checks, cell delays and net delays. This option should be used when the SDF file is intended for simulator use. Using this option leads to inaccurate delay estimation in PrimeTime, so the user should use caution with this option.

`-no_edge_merging` *`arc_type_list`*

Specifies a list of arc types that are not to be compressed in the SDF file through edge merging. When this option is not selected PrimeTime merges the posedge/negedge arcs only if they are identical to within a tolerance of 0.00001 times the absolute value of the smaller delay.

Allowed values are one or more of the following.

- *timing_checks* - Indicates that timing checks are not to be compressed in the SDF file through edge merging.
- *cell_delays* - Indicates that cell delays are not to be compressed in the SDF file through edge merging.

`-delay_calculation_only_mode`

This option is intended for speeding-up flows that use PrimeTime for delay calculation. The *write_sdf* command first checks if the design is in the updated state. If it is, it simply writes the already computed delays and leaves the design in the updated state, regardless of the presence of the *-delay_calculation_only_mode* option. If the design is not in the updated state, by default, the command triggers the *update_timing* command, which brings the design in the updated state. However, if the *-delay_calculation_only_mode* option is used, PrimeTime does not trigger the *update_timing* command; it triggers only the operations that are necessary for writing the SDF file; the design is left in the not-updated state.

`-exclude_cells` *`cell_list`*

Specifies a list of cells that are not to be included in the SDF file. The net delays of the excluded cell's boundary pins are not excluded.

`-no_derates`

Specifies not to apply a derate to any delay in the generated SDF file.

`-objects`

Specifies that only cells that have the user-defined attribute *is_write_sdf_object* set to *true* will be included in the SDF file. Net arcs will be included in the SDF if they meet at least one of the following three conditions:

- The net arc is between two cells that both have attribute *is_write_sdf_object* set to *true*.
- The net arc is between a cell that has attribute *is_write_sdf_object* set to *true* and a port.
- The net arc belongs to a net that has attribute *is_write_sdf_object* set to *true*.

`-sms_scenarios` *`sms_scenarios_list`*

Specifies a collection of SMS scenarios (from the *get_sms_scenarios* command) to analyze. The worst (min/max) delays across the specified SMS scenarios are printed. This option is only available with SMVA or SMC analysis.

`-all_scenarios`

Automatically creates a separate SDF file for every fully different voltage configuration in SMVA analysis. All possible voltage configuration combinations are captured. This option is only available with SMVA analysis.

`file_name`

Specifies the name of the SDF file to be written.

Description

This command writes leaf cell pin-to-pin timing information to a disk file. Timing information is written in SDF format using versions v2.1 or v3.0. The timing file contains data associated with the netlist from which it is created.

By default, the file is written as simple text; you can use the *-compress* option to compress the file. In some cases, if you omit the file extension, the *write_sdf* command appends it to *file_name*. For example, for gzip formatted files, the *write_sdf* command appends .gz to the file name if .gz was not specified.

Use the *write_sdf* command only when the instance names in the design agree with the naming conventions of the system to which the timing file is written.

Timing information can be written in multiples of ns, ps, and us. The time unit in the SDF file is that specified in the logic library, and is written in the timing file under "timescale". If there is no time unit in the library, the unit is assumed to be a multiple of nanoseconds.

For efficiency, PrimeTime merges rise/fall delay triplets in SDF, if they are identical to within a very small tolerance.

Examples

The following example writes timing information for the *MULT16* design to a disk file called mult16.sdf, using SDF version 2.1:

```
pt_shell> write_sdf -version 2.1 mult16.sdf
```

You can use the *-level* option to write out interblock net delays. Assume a design with three hierarchical cells, h, h/h1, and h/h2; and two leaf cells, h/h1/u1 and h/h1/u2. The following command writes out all arcs that begin or end inside h but outside h/h1 and h/h2:

```
pt_shell> write_sdf -levels 1 -instance h h.sdf
```

The following example writes out the same arcs as the previous example, plus arcs that cross the boundary of h. Note that either the from or the to pin must be outside h and the other inside h. Therefore, the command does not write a feedthrough net through h on this particular design. Similarly, a net arc representing net h/h1/u1/Y -> h/h1/out -> h/h1/in -> h/h1/u2/A is not written out, because both pins h/h1/u1/Y and h/h1/u2/A are inside h/h1; that is, both are outside the region defined by the *-level 1 -instance h* option.

Feedback

```
pt_shell> write_sdf -levels 1 -instance h -input_port_nets \\
            -output_port_nets h.sdf
```

See Also

- read_sdf
- remove_annotated_check
- remove_annotated_delay
- report_annotated_check
- report_annotated_delay
- set_annotated_check
- set_annotated_delay

---

write_session_settings

Writes current session settings to an encrypted file, which can be later sourced with the *source* command.

Syntax

status *write_session_settings*

`-output` *file_name*

Data Types

file_name	string

Arguments

`-output` *file_name*

Writes current session settings to the specified file.

Description

The *write_session_settings* command writes current session settings to a file in Tcl byte code (TBC) binary format. The session settings include application variables and other design-independent settings. This enables porting settings from one session to another. For example, you can write out the settings from one session and source the file in the other session to restore the same settings.

Note that the *write_session_settings* command does not write any design-specific settings such as design data and constraints. Those can be saved by the *save_session* command.

Feedback

Examples

The following example changes the *timing_crpr_threshold_ps* variable setting from the default value to 10 and writes it to a file along with other variable settings.

```
pt_shell> set_app_var timing_crpr_threshold_ps 10
  ...
pt_shell> write_session_settings -output my_session.tbc
```

In a new PrimeTime session, the following example sources the my_session.tbc file. After sourcing, the *timing_crpr_threshold_ps* application variable has changed from the default value to 10.

```
pt_shell> source my_session.tbc
pt_shell> printvar timing_crpr_threshold_ps
timing_crpr_threshold_ps = "10"
```

See Also

- save_session
- source

---

write_spice_deck

Writes out a SPICE deck for the timing paths or arcs collected by the *get_timing_paths* or *get_timing_arcs* command.

Syntax

status *write_spice_deck*

```
[-align_aggressors]
[-analysis_type type]
[-ground_coupling_capacitors]
[-header header_file_name]
[-initial_delay delay]
[-logic_one_name v1name]
[-logic_one_voltage v1]
[-logic_zero_name v0name]
[-logic_zero_voltage v0]
[-minimum_transition_time trans]
[-output file_name]
[-sub_circuit_file spice_sub_circuit_file]
[-sweep_exhaustive]
[-sweep_size number_of_points]
[-sweep_step num]
[-sample_size number_of_samples]
[-time_precision precision]
[-transient_size tran_size]
[-transient_step tran_step]
```

```
[-use_probe]
[-user_measures user_measure_list]
[-variation]
[-mc_sample_size number_of_mc_samples]
paths_arcs_list
```

Data Types

type	list
header_file_name	string
delay	float
v1name	string
v1	float
v0name	strong
v0	float
trans	float
file_name	string
spice_sub_circuit_file	string
number_of_points	unsigned
num	float
number_of_samples	integer
precision	unsigned
tran_size	float
tran_step	float
user_measure_list	list
number_of_mc_samples	integer
paths_arcs_list	collection

Arguments

`-align_aggressors`

Aligns multiple aggressor transitions to apply the worst-case crosstalk effects on each victim net for the conditions specified by the *-analysis_type* option. The *-align_aggressors* option applies only to a *net* timing arc (gathered by the *get_timing_arcs* command) with an annotated RC network.

The relative switching times of active aggressors on the net arc computed by crosstalk delay or noise analysis are written out in the corresponding piecewise linear (PWL) statement. The victim net switches after the initial delay specified by the *-initial_delay* option and the active aggressors switch relative to that. The tool uses the calculation engine to get the worst-case alignment and uses a similar setup so that the relative alignment is valid. For example, filtered aggressors are not considered effective during calculation and so their coupling capacitors are grounded.

`-analysis_type` *type*

Specifies the type of crosstalk or noise analysis performed in the SPICE deck for a timing arc:

- Crosstalk: *max_rise* (the default), *max_fall*, *min_rise*, or *min_fall*
- Noise: *above_high*, *above_low*, *below_high*, or *below_low*

This option applies only to a timing arc, not a timing path.

`-ground_coupling_capacitors`

Prevents crosstalk analysis in the SPICE simulation. Coupling capacitors are connected to ground with a factor of 1, and aggressors of the timing path or timing arc are not written to the SPICE deck.

`-header` *header_file_name*

Specifies the path to the user header file. The file content is copied to the generated SPICE deck. You can use this file to identify the SPICE deck, to include library files, or for any other purposes to support the SPICE run.

`-initial_delay` *delay*

Specifies the initial delay, in library time units, added to all PWL statements. The default is the longest clock period, or 1.0 library unit for asynchronous designs.

Setting the delay to zero makes generating a ramp difficult and is therefore not recommended.

`-logic_one_name` *v1name*

Specifies the name of the upper rail voltage source. The default is VDD.

`-logic_one_voltage` *v1*

Specifies the upper rail voltage of the gate input pins. This is used in the PWL and voltage sources generated by the command. The default is main library voltage.

`-logic_zero_name` *v0name*

Specifies name of the default lower rail voltage source. The default is VSS.

`-logic_zero_voltage` *v0*

Specifies the lower rail voltage of the gate input pins. The default is 0 volts.

`-minimum_transition_time` *trans*

Specifies the minimum transition time in nanoseconds (ns), which is used in generated PWL transitions where the transition time computed by the PrimeTime tool is less than the specified minimum value. The default is 0.001 ns. Transition times less than 0.0001 ns are not recommended.

`-output` *name*

Specifies the name of the SPICE deck file written by the command. If multiple files are generated, they are named by appending characters to the specified name. If the *-sample_size* option is used, the *-output* option specifies the name of the directory created to hold the generated SPICE deck files.

If you do not specify an output name, the command writes the SPICE deck to the screen.

`-sub_circuit_file` *spice_sub_circuit_file*

Specifies the path to the file that contains all the SPICE .subckt definitions of logic gates in the timing paths. By default, a subcircuit call uses the pin order in the Synopsys .lib file. Use this option if the SPICE subcircuit has a different pin order from that of the .lib file.

`-sweep_exhaustive`

If the *-sample_size* option and *-align_aggressors* option are set, then by default, sweeping is done for all aggressors in the same way to avoid excessive simulation runtime. If this option is set, sweeping is done in an exhaustive manner.

With exhaustive sweeping, the number of SPICE simulations grows geometrically as the number of active aggressors increases. For example, using *-sweep_exhaustive -sample_size 3* for five active aggressors results in 3^5 (243) simulations.

`-sweep_size` *number_of_points*

Specifies the number of sweep points generated for each active aggressor of the net arc. By default, the sweep table size is equal to the specified number of points.

With the *-sweep_exhaustive* option, the number of simulations increases geometrically with the number of the active aggressors.

The specified number of points can be no more than 20 the *-sweep_exhaustive* option is set.

Use this option with the *-align_aggressors* option.

`-sweep_step` *num*

Specifies the maximum time interval between sweep points generated for each active aggressor of the net arc. The unit size is nanoseconds. The default is 0.1 ns. Use this option with the *-align_aggressors* and *-sweep_size* options.

`-sample_size` *`number_of_samples`*

Specifies the number of SPICE deck files to be created for variation-aware timing analysis. This option takes the name of the directory specified by the *-output* option and creates multiple SPICE deck files that correspond to various samples of the variations defined.

`-time_precision` *`precision`*

Specifies the number of digits written out for time values in the PWL statements. The default is 6. The allowed range is from 1 to 20.

`-transient_size` *`tran_size`*

Specifies the total transient time used in the SPICE .tran statement. The unit size is nanoseconds.

`-transient_step` *`tran_size`*

Specifies the transient step size used in the SPICE .tran statement. The unit size is nanoseconds. The default is 0.001 ns.

`-use_probe`

Uses the .probe statement to output the node voltage instead of the .print statement.

`-user_measures` *`user_measure_list`*

Uses the user-specified measures instead of the ones generated automatically by the SPICE deck. To remove all automatically generated .measure and .print statements from the SPICE deck, specify *-user_measures {}*.

`-variation`

Generates a SPICE deck for Monte Carlo based simulation for path arrival. The number of Monte Carlo samples follows the *-mc_sample_size* option. This option only works for a timing path, not a timing arc.

`-mc_sample_size` *`number_of_mc_samples`*

Specifies the number of Monte Carlo samples in the SPICE deck. By default, *number_of_mc_samples* is 5000. This option has impact only with the *-variation* option.

`paths_arcs_list`

Specifies a collection of timing paths gathered by the *get_timing_paths* command or a timing arc collection gathered by the *get_timing_arcs* command. A SPICE deck is generated for each path or arc in the collection.

Feedback

Description

This command writes out SPICE deck files, one file for each timing path or timing arc from a collection generated by the *get_timing_paths* or *get_timing_arcs* command. The SPICE deck file contains the circuit elements, parasitic components, and side pin voltages of the path or arc. For a timing arc, the SPICE deck can include simulation of crosstalk effects through parasitic capacitors with worst-case alignment of aggressor transitions.

Using the generated data files and a SPICE circuit simulator such as HSPICE, you can run a simulations of signal transitions propagated through the timing path or timing arc. The simulation results confirm the accuracy of PrimeTime static timing analysis.

The names of multiple SPICE deck files are based on the string provided by the *-output* option. For example, if you specify *-output timing_path.spo*, the first file name is timing_path.spo, the second is timing_path_00001.spo, the third is timing_path_00002.spo, and so on.

The *write_spice_deck* command also puts the corresponding stimulus data, such as PWL and PULSE statements, in separate stimulus files. The example, the stimulus file names can be timing_path_stim.spo, timing_path_00001_stim.spo, and so on. A stimulus file is included in the main circuit file by means of the SPICE .include statement.

If you specify a subcircuit file using the *-sub_circuit_file* option, all lines except .include and .subckt are ignored. The file on the .include line is further opened and parsed for other .include and .subckt lines. The pin order of each .subckt line is recorded and applied when a subcircuit call is generated. If a gate used is not in the subcircuit file, the .lib pin order is used. A warning is generated if a pin in the subcircuit definition is not found in the .lib file. This mismatched pin is written out to the SPICE deck in the same order as it appears in the subcircuit file. An error message is generated when a .lib gate pin is not found in the subcircuit definition.

Note that side input pins are sensitized locally.

The timing PWL voltage sources are defined by the *set_library_driver_waveform* command. The default is the standard Synopsys predriver waveform. The *set_library_driver_waveform* command defines the waveform shape used to characterize the timing library. If advanced waveform propagation is enabled and a distorted waveform is available, the actual waveform is used to define the shape of the PWL voltage sources instead. If you want to use the library waveform shape, use the *set_annotated_transition* command to apply the transition slew value on the pin.

The *-align_aggressors* option applies only to active aggressors of net timing arcs. The inactive or filtered aggressors are set to be quiet and their coupling capacitors grounded. To report the active aggressors of a victim, use *report_delay_calculation -crosstalk* or *report_noise_calculation*.

While the *-aligned_aggressors* option is set, the *-sweep_size* option set to more than zero sweeping points writes out the PWL in sweep form. A third file with the sweep data is

generated, for example named timing_arc_sweep.spo, which is also included in the main circuit file by means of the SPICE .include statement

Under most circumstances, the *-align_aggressors* option writes out the alignment to produce the worst-case stage delay. However, under some corner cases, for example when the crosstalk bump is very large, the *-sweep_size* option should be used in conjunction with *-align_aggressors* to obtain the worst-case alignment.

The sweep points center around the aligned aggressor switching time generated without the *-sweep_size* option. If an even number is given, one more sweep point on the smaller switch time side is generated. The time difference between the sweep points is controlled by the *-sweep_step* and *-sweep_size* options, and the aggressor sweep range is computed by the crosstalk delay engine. The command uses the lesser of the (sweep_range)/(number_of_sweep_points-1) for each aggressor or the *-sweep_step* option.

Use the *-user_measures* option to create specific measure statements instead of automatically generating them. The *user_measure_list* argument is a list of user-defined measures. Each user-defined measure can be one of the following types.

```
{delay from_pin/port_name to_pin/port_name [meas_name]}
{slew pin_or_port_name [meas_name]}
{noise_peak pin_or_port_name [meas_name]}
{user_string "user_defined_string"}
```

The *write_spice_deck* command uses these user-defined measure instructions and creates SPICE .measure statements with proper trip points and switching directions. The SPICE measures are in the same order as the measure generated when using the *user_measure_list* option. When this option is provided, the user-defined measure overrides the automatic .measure and .print statements.

You can specify zero or more measures in the list. When there is no user-defined measure in the list, for example, using *-user_measures { }*, the SPICE deck does not have any kind of .measure or .print statements. You can also insert user-specific measures in the user-defined string, which is inserted verbatim. For the *delay*, *slew*, and *noise_peak* types, you can give a name to the measurement as *meas_name*. If it is missing, the *write_spice_deck* command automatically generates a name for the measurement. A user-defined measure must match with one of the previous formats.

If you specify the *-sample_size* option while performing variation-aware timing analysis, the command creates a directory and writes multiple SPICE deck files inside it.

To use the *write_spice_deck* command, you must have a PrimeTime SI license.

Examples

The following example writes a SPICE deck for timing paths gathered by the *get_timing_paths* command, using the my_subckt.sp file as the source of all SPICE .subckt definitions of all gates in the timing path.

Feedback

```
pt_shell> write_spice_deck -output path.spo \\
            -sub_circuit_file my_subckt.sp [get_timing_paths]
```

The following example writes a SPICE deck for a timing arc gathered by the *get_timing_arcs* command.

```
pt_shell> write_spice_deck -output path.spo \\
            -analysis_type above_low \\
            -sub_circuit_file my_subckt.sp [get_timing_arcs -from U1/A]
```

The following example writes a SPICE deck for a timing arc, including crosstalk with sweep analysis to find the worst aggressor alignment.

```
pt_shell> write_spice_deck -output ./arc.spi \\
            -analysis_type max_rise \\
            -align_aggressors -sweep_size 3 -sweep_step 0.01 \\
              [get_timing_arcs -from I1_1/Z -to I1_2/A ]
```

The following example generates measure statements to measure the slew at the path output and delay from input to output.

```
pt_shell> write_spice_deck [get_timing_paths -from [get_port i2] \\
            -to [get_port o2]] -user_measures { {slew o2 my_out_slew } \\
            {delay i2 o2 path_delay} {user_string ".print o2"} }
```

```
.measure tran my_out_slew
+ trig v(o2) val = 2.16 td = 7.5e-09 fall = 1
+ targ v(o2) val = 0.54 td = 7.5e-09 fall = 1
.measure tran path_delay
+ trig v(i2) val = 1.35 td = 7.5e-09 fall = 1
+ targ v(o2) val = 1.35 td = 7.5e-09 fall = 1
.print o2
```

See Also

- get_timing_arcs
- get_timing_paths
- report_delay_calculation
- report_noise_calculation
- set_library_driver_waveform

---

write_training_data

Writes out training data recorded after the *record_training_data* command for faster power recovery in future sessions using similar design data.

Feedback

Syntax

status *write_training_data*

```
-output file_name
[-force]
```

Data Types

```
file_name     string
```

Arguments

`-output file_name`

Specifies the name of the file to use for writing out the training data. You can specify an absolute or relative path the to file. Specifying a relative path writes out the training data to the current working directory.

To invoke accelerated ECO power recovery in future sessions, specify the training data file using the *-training_data* option of the *fix_eco_power* command.

`-force`

Overwrites the specified file if the file already exists. Without this option, the command does not save the training data if the specified file already exists.

Description

The *write_training_data* command writes training data to a file after you run the *record_training_data* command and perform cell sizing operations for power recovery using the *size_cell* commands.

In future PrimeTime sessions, you can invoke the same cell sizing operations during ECO power recovery for cells in the same or similar design context. When you use the *fix_eco_power* command, use the *-training_data* option to specify the names of the files previously generated by the *write_training_data* command, or use the *training_data_directory* variable to specify the directory where the training data files can be found.

In this flow, the PrimeTime tool does not check or evaluate the quality or benefit of the *size_cell* commands. It simply records the cell sizing changes and saves the timing and topological conditions surrounding those changes for future use during ECO power recovery.

Examples

The following example starts power optimization training, performs cell sizing for power recovery, and writes the session training data into a file in a specified absolute path.

```
pt_shell> record_training_data
pt_shell> size_cell U100 ...
```

```
 ...
pt_shell> source my_eco_changes.tcl  # Cell sizing for power recovery
 ...
pt_shell> size_cell U200 ...
 ...
pt_shell> write_training_data -output /home/train_dir/my_training.td
```

In a future PrimeTime session using the same design or a similar design, the following command invokes the same cell sizing operations for power optimization using the previously saved training data.

```
pt_shell> fix_eco_power -training_data /home/train_dir/my_training.td ...
```

Alternatively, use the *training_data_directory* variable to specify the directory where the training data files can be found:

```
pt_shell> set_app_var training_data_directory /home/train_dir
/home/train_dir
pt_shell> fix_eco_power ...
```

See Also

- fix_eco_power
- record_training_data

---

write_tsv_timing

Generate TSV feedthrough path timing data at the die(block) level.

Syntax

```
int write_tsv_timing
  [-corner_id id]
  [-output file_name]
```

Data Types

id	int
file_name	string

Arguments

`-corner_id` *id*

Specifies the corner id in integer.

`-output` *filename*

Specify the name of the tsv path timing data file to be written.

Feedback

Description

The *write_tsv_timing* command writes out a tsv feedthrough path timing data file in binary format at the die level for the specified corner.

Note: update_timing -full needs to be done before you use this command.

Examples

The following example writes a tsv feedthrough path timing data file tsv.dat at the die level for the specified corner 1.

```
prompt> write_tsv_timing -corner_id 1 -output tsv.dat
```

See Also

- read_tsv_timing

---

write_vectors

Generate vectors in form of a VCD file.

Syntax

status *write_vectors*

```
[-format file_format]
file_name
```

Data Types

```
file_format          string
file_name            string
```

Arguments

file_format

Specifies the file format for writing out vectors. Supported format is VCD/vcd or FSDB/fsdb. Default is VCD.

file_name

Specifies the name of the file to which the VCD is to be written.

Description

The *write_vectors* command writes events generated from vector free vector generation into a VCD or FSDB file. By default, a RTL VCD file is written out, which means only events on synthesis invariant points are written out.

The command *set_vector_generation_options* is used to set options on how vectors are generated. There are three different types of vectors can be generated, and each one can be specified by corresponding option in the command *set_vector_generation_options*. Default is single clock cycle vector generation. The second one is multi-cycle vector generation, which is enabled by the option *-number_of_cycles*. The third one is special mode vector generation for single cycle, which is enabled by the option *-special_nets*.

Examples

The following example writes a RTL VCD file after average power analysis:

```
pt_shell> set_power_analysis_options -through_mode
pt_shell> update_power
pt_shell> write_vectors rtl.vcd
```

The following example sets a clock gating percentage 30% rather than default 50%, and then writes a RTL VCD file:

```
pt_shell> set_power_analysis_options -through_mode
pt_shell> update_power
pt_shell> set_clock_gating_percentage -value 0.3
pt_shell> write_vectors rtl.vcd
```

The following example writes a RTL FSDB file after average power analysis:

```
pt_shell> set_power_analysis_options -through_mode
pt_shell> update_power
pt_shell> write_vectors -format FSDB rtl.fsdb
```

The following example reads patterns from a VCD file for all control nets, and sets one hot mode for a list of bus nets. For all primary input, constraint mode is applied. Then the command *write_vectors* is invoked to generate 100 cycle VCD.

```
pt_shell> set_vector_generation_options -number_of_cycles 100
pt_shell> set_net_pattern -file control.vcd -format VCD -strip_path tb
 -object_list [get_nets control*]
pt_shell> set_net_pattern -mode one_hot -object_list [get_nets mbus[*]]
pt_shell> set_net_pattern -object_list [get_nets -of [get_ports]] -mode
 constraint
pt_shell> write_vectors 100_cycle.vcd
```

The following example shows how to generate a special mode VCD file, where all nets are considered as special nets, and 75% of those nets are toggling. At the same time, state 0 is set on all clock gating cell enable nets through command *set_net_pattern*.

```
pt_shell> set_vector_generation_options -special_nets [get_nets -hier]
pt_shell> set_vector_generation_options -toggle_percentage 0.75
pt_shell> set_net_pattern -pattern 0 -object_list [get_nets -of [get_pins
 icg_*/enble]
pt_shell> write_vectors special_mode_75.vcd
```

Feedback

See Also

- set_vector_generation_options
- set_memory_percentage
- set_clock_gating_percentage
- set_net_pattern

---

write_waiver

Writes existing waivers into an output file. The file can be sourced to restore waivers in a new session.

Syntax

Boolean *write_waiver*

```
-output output_file
[-force]
```

Data Types

```
output_file     string
```

Arguments

`-output output_file`

Writes all existing waivers into the given output file.

`-force`

Overwrites the output file if it already exists.

Description

This command is available only if you invoke the pt_shell with the *-constraints* option.

Writes existing waivers into an output waiver file. To restore waivers from the *write_waiver* output, you need to source the waiver file in each scenario. Sourcing a waiver file in one scenario restores only the waivers active in that scenario, and waivers for scenario-independent rules.

See Also

- create_waiver
- remove_waiver

Feedback

write_xdomain_design

Writes a cross-domain design that consumes less memory to perform SMVA than the original design.

Syntax

status *write_xdomain_design*

[*dir_name*]

Data Types

dir_name	string

Arguments

dir_name

Specifies a directory name in which a cross-domain design is created. The specified directory contains multiple files after creation.

If you do not specify a directory name, the default is *xdomain_design_directory*.

Description

This command reduces hardware requirements by creating a cross-domain design that requires less memory to perform SMVA. A *cross-domain design* is a small subset of the original design that contains only paths crossing voltage domains. For example, it retains only cells and nets on cross-domain paths and omits cells and nets on non cross-domain paths. Special timing and crosstalk models are created for disconnected nets and pins in the cross-domain design so that timing quality of results (QoR) in the cross-domain design is very close to the timing QoR of the original design. Its goal is to have timing QoR close enough to produce similar SMVA run for path crossing domains compared to the cross-domain design generated from the original design.

Memory Reduction

After a cross-domain design is written, memory reduction may not be proportional to the number of reduced cells because PrimeTime's memory consumption also includes fixed memory overhead like loading library. For example, a design loaded with a large number of larger libraries shows lower memory reduction than a design loaded with one small library.

PrimeTime shows an estimated design reduction summary after the *write_xdomain_design* command finishes. Note that the numbers are not exact but estimated values. The purpose of the estimate is to provide rough numbers for design reduction. Exact cell, net, and pin counts as well as peak memory will be available when the *read_xdomain_design* command finishes.

Feedback

Examples

This section describes an example to convert existing SMVA scripts to new scripts utilizing the *write_xdomain_design* and *read_xdomain_design* commands.

Example Scripts

The following example is a typical script for STA and SMVA combined together.

```
################################################
# FILE: run_sta_and_write_xdomain_design.tcl
#   To run STA and write_xdomain_design.
#   Runs in 100 GB machine.
################################################
#
# STA
#
set timing_enable_cross_voltage_domain_analysis true
set timing_cross_voltage_domain_analysis_mode capture_reduced_model
read_verilog...
read_parasitics...
read_sdf...
update_timing...
report_timing...
#
# Creates an xdomain design
#
write_xdomain_design
exit

################################################
# FILE: run_read_xdomain_design_and_xdomain.tcl
#   To run cross-domain.
#   Runs in 20 GB machine.
################################################
#
# Read the cross-domain design
#
set timing_cross_voltage_domain_analysis_mode full
read_xdomain_design

update_timing...
report_timing...
exit
```

See Also

- read_xdomain_design