StarRC™ User Guide and Command Reference

Version V-2023.12-SP4, June 2024

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

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

Disclaimer

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

Trademarks

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

Free and Open-Source Licensing Notices

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

Third-Party Links

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

www.synopsys.com

Feedback

Contents

About This Manual

This user guide describes how to use the StarRC tool to perform parasitic extraction and to use the grdgenxo tool to model advanced processes.

This manual is intended for circuit design and layout engineers who need to analyze the effects of parasitic capacitance and resistance on advanced circuit designs.

This preface includes the following sections:

New in This Release

Information about new features, enhancements, and changes, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the StarRC Release Notes on the SolvNetPlus site.

Related Products, Publications, and Trademarks

For additional information about the StarRC tool, see the documentation on the Synopsys SolvNetPlus support site at the following address:

https://solvnetplus.synopsys.com

You might also want to see the documentation for the following related Synopsys products:

- Parasitic Explorer
- PrimeTime® Suite
- IC Compiler™
- IC Compiler™ II
- Fusion Compiler™
- Custom Compiler™
- IC Validator

Feedback

- Raphael™
- CustomSim™

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates syntax, such as `write_file`
`Courier italic`	Indicates a user-defined value in syntax, such as `write_file` *`design_list`*
`Courier bold`	Indicates user input—text you type verbatim—in examples, such as `prompt> write_file top`
`Purple`	• Within an example, indicates information of special interest. • Within a command-syntax section, indicates a default, such as `include_enclosing = true | false`
[ ]	Denotes optional arguments in syntax, such as `write_file [-format fmt]`
...	Indicates that arguments can be repeated as many times as needed, such as *`pin1 pin2 ... pinN`*.
|	Indicates a choice among alternatives, such as `low | medium | high`
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Bold	Indicates a graphical user interface (GUI) element that has an action associated with it.
Edit > Copy	Indicates a path to a menu command, such as opening the **Edit** menu and choosing **Copy**.
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.

Feedback

Customer Support

Customer support is available through SolvNetPlus.

Accessing SolvNetPlus

The SolvNetPlus site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNetPlus site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNetPlus site, go to the following address:

https://solvnetplus.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNetPlus site, click REGISTRATION HELP in the top-right menu bar.

Contacting Customer Support

To contact Customer Support, go to https://solvnetplus.synopsys.com.

Part 1: StarRC User Guide

1

Introduction to StarRC

The StarRC tool extracts parasitics such as resistors, capacitors, and inductors from databases that represent integrated circuit layout designs. You can use the StarRC tool to generate netlists for many types of analysis, such as timing, noise, and electromigration.

For more information, see the following topics:

- StarRC Features
- StarRC Licensing
- StarRC Documentation

StarRC Features

The StarRC parasitic extraction tool is part of the Synopsys signoff analysis suite. The tool provides a flexible framework that efficiently and accurately covers the entire extraction spectrum, whether analyzing a full chip containing millions of transistors or examining a few critical nets in three-dimensional field solver mode. You can use a variety of input data formats and specify many options for customizing the output.

Some of the features of the StarRC tool are as follows:

- Full-chip parasitic extraction for use with noise, electromigration, and timing verification tools
- Integrated field solver for accurate three-dimensional analysis of selected nets
- Ability to use characterization files provided by major foundries for process development kits
- Both gate-level and transistor-level extraction
- Both flat and hierarchical extraction
- Interoperability with many Synopsys and third-party layout-versus-schematic, custom layout, timing analysis, and simulation tools
- Ability to analyze early-stage designs that have opens, shorts, and other design rule violations

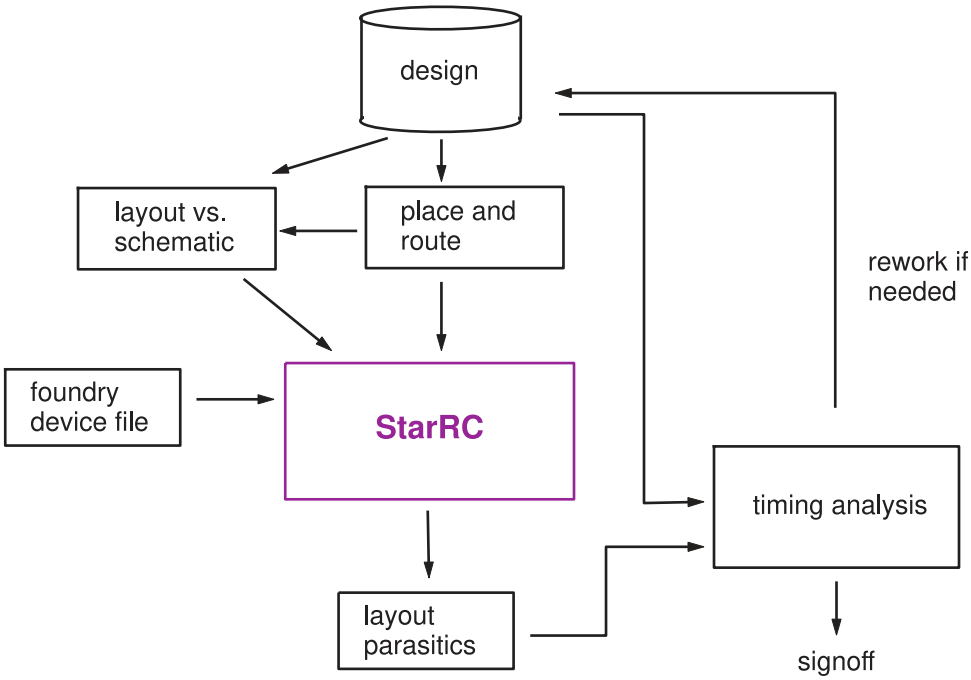
StarRC Ultra features offer advanced capabilities, including:

- Efficient distributed processing and netlist creation techniques to optimize runtime and memory usage for very large designs
- A fast Engineering Change Order flow that streamlines extraction by analyzing only those nets that have been changed from the reference design
- Analysis of FinFET and other advanced transistor designs
- Extraction for double or multiple patterning processes
- Ability to create customized characterization files using comprehensive process modeling for interconnect layers
- Simultaneous multicorner analysis, which saves runtime by examining related corners in parallel

Feedback

Figure 1 shows how StarRC extraction is used in a typical design flow.

Figure 1 StarRC Extraction in the Design Flow



After a design layout is complete, the circuit timing must be tested. Accurate timing analysis requires that all of the parasitic resistances and capacitances are taken into account. The StarRC tool uses the chip layout along with the process description (usually obtained from a foundry) to extract millions of parasitic devices. If the design must be modified to fix timing violations, the extracted parasitics must also be updated. Extracted parasitics are also important for other tools such as circuit simulators and electromigration analysis tools.

StarRC Licensing

The following licenses govern StarRC tool use, in order from basic feature coverage to most advanced feature coverage:

- STAR-RC2_MANAGER (known as the StarRC license)
- STAR-RC2_ULTRA_MANAGER (known as the Ultra license)
- STAR-RC2_AGP (known as the AG+ license)
- STAR-RC2_AG3 (known as the AG3 license)
- STAR-RC2_ULTRAPLUS (known as the Ultra+Custom license)
- STAR-RC2_ULTRAPLUS_DIGITAL (known as the Ultra+Digital license)

Figure 2 illustrates the license checking procedure for features covered by the Custom, StarRC, and Ultra licenses. When a StarRC run begins, the tool checks for available licenses based on the features specified in the command file. Multiple lower-tier licenses can be used to run upper-tier features. For example, two StarRC license keys or four Custom license keys can enable a run with an Ultra feature.

For each feature type, when a valid license configuration is found, the run starts. If a valid license configuration is not available, the run never starts, unless you enable license queuing. If license queuing is enabled, the tool waits for a short period of time, then checks again for available licenses in the order shown in Figure 2. To enable license queuing, set the `STARRC_LICENSE_WAIT` environment variable, as follows:

```
% setenv STARRC_LICENSE_WAIT yes
```

If you use the `-custom` option with the `StarXtract` command, the tool checks only for Custom licenses. If you use the `-ultra` option, the tool checks only for Ultra licenses.

The following usage notes apply:

- When the number of available licenses is less than the number of cores specified in the `NUM_CORES` command and the `STARRC_LICENSE_WAIT` environment variable is set to `YES`, the StarRC tool waits until enough licenses are available to proceed.
- If you want to start a job immediately but the number of available licenses is less than the number of cores specified in the `NUM_CORES` command, set the `STARRC_LICENSE_WAIT` environment variable to `NO` or delete it (the default is `NO`). In this case, the job starts immediately using the available licenses and fewer cores than the number specified in the `NUM_CORES` command.

Feedback

When you use set the STARRC_LICENSE_WAIT environment variable to YES, the tool checks out licenses in the following order:

1. STAR-RC2_AGP (known as the AG+ license)
2. STAR-RC2_AG3 (known as the AG3 license)
3. STAR-RC2_ULTRAPLUS (known as the Ultra+Custom license)
4. STAR-RC2_ULTRAPLUS_DIGITAL (known as the Ultra+Digital license)

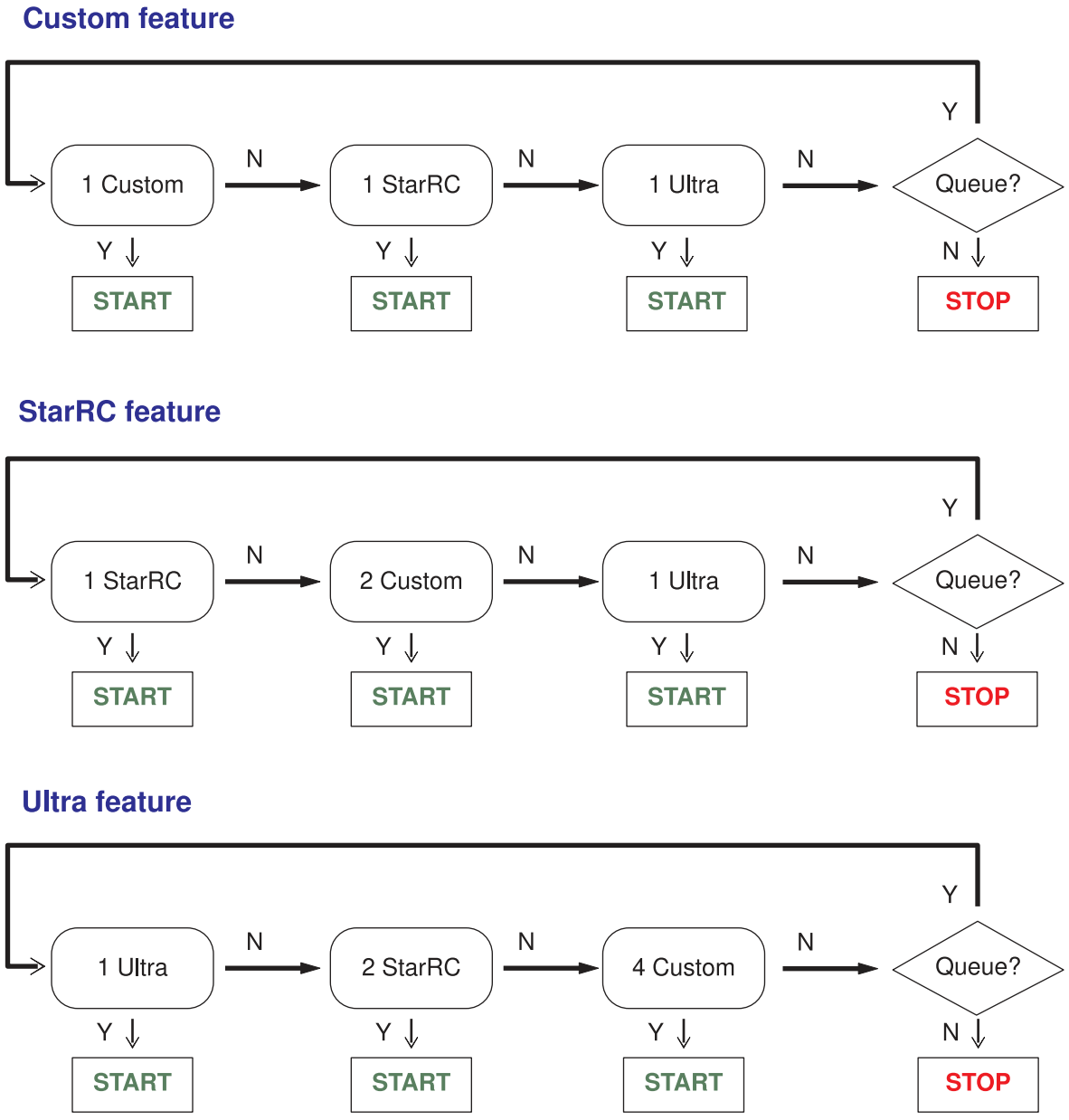
Table 1 lists the behavior of the StarRC tool based on the following settings of the STARRC_LICENSE_WAIT environment variable:

Table 1 Behavior of the StarRC Tool With the STARRC_LICENSE_WAIT Environment Variable Settings

Settings	Behavior
Not set	Issues an error message if a license key cannot be checked out.
Set but the field is empty	Waits indefinitely until a license key is available to check out.
Set as string	Waits indefinitely until a license key is available to check out.
Set to a numeric value	Defines a timeout based on a numerical value, and issues an error message if the key cannot be checked out within timeout. `timeout = N` Where N is rounded to the nearest multiple of 5.

Feedback

Figure 2 License Checking Procedure



Feedback

Licensing Requirements for Advanced Process Nodes

The StarRC AG+ license enables the use of features for process nodes 5 nm or smaller. License requirements are as follows:

- To check out an AG+ license, an Ultra or Ultra+license is also required. Combinations of standard or custom licenses cannot be used in place of Ultra or Ultra+ licenses.
- The combination of one AG+ license and one Ultra or Ultra+ license enables extraction on up to 4 cores (specified by setting the `NUM_CORES` command). If you run on 8 cores, you need two AG+ licenses and two Ultra or Ultra+ licenses, and so on.
- If the nxtgrd file does not contain a process header, an AG+ license is required if any of the following conditions exist:
  - The `WMIN` value for the gate conductor layer (indicated by `LAYER_TYPE = GATE`) is less than or equal to 5 nm.
  - There is no conductor layer defined with the `LAYER_TYPE = GATE` statement and the `WMIN` value for any layer is less than or equal to 5 nm.
- If the nxtgrd file contains a process header, an AG+ license is required if any of the following conditions exist:
  - The `PROCESS_NODE` value is less than or equal to 5.0.
  - The `PROCESS_TYPE` statement is set to any of the following values: `VFET`, `VTFET`, or `GAA`. Additional values can be defined at any time to specify new process technologies, which might also require the AG+ license.

Note:

A process header is a section in the nxtgrd file that includes the `PROCESS_NODE`, `PROCESS_VERSION`, `PROCESS_FOUNDRY`, `PROCESS_TYPE`, and `PROCESS_CORNER` statements. A process header is required for all nxtgrd files that contain a `MULTIGATE` statement or a `WMIN` value less than or equal to 8 nm for any conductor layer.

Feedback

StarRC Documentation

If you need help, information is available from the following resources:

- This user guide
- Message man pages displayed with the `starrc_man` command
- SolvNet articles

See also the following topics:

- StarRC Formal Messages
- Computing Environment Warning Messages
- StarRC Message Man Pages
- Error Message Control

StarRC Formal Messages

StarRC formal messages indicate that a condition requires user attention. The severity levels are as follows:

- Information: No action required if the condition is acceptable

  For example, the SX-1429 message states that pin information from a DEF macro is used because there is no corresponding LEF macro. This scenario might be expected and acceptable, and the tool has enough information to proceed with extraction. However, you should review all information messages to ensure that you understand their implications.

- Warning: Unexpected condition, but execution does not stop

  For example, the SX-1505 message states that a layer specified in a skip cell layers file is not found in the design database. This scenario might be expected and acceptable, and the tool has enough information to proceed with extraction. However, the message might also indicate an oversight in the design or command file that should be investigated. Only you can determine if the condition is acceptable.

- Error: Serious condition, likely to be undesirable, but execution does not stop

  For example, the SX-2780 message describes an issue with a Milkyway library.

- Severe: Serious condition that halts execution, but only after the current operation is complete, to allow the reporting of similar errors

  For example, the SX-1837 message states that a database via layer is not mapped to an ITF via layer. Extraction cannot proceed, but the tool completes the analysis of the layer mapping file to find similar errors before terminating the run.

- Fatal: Serious condition that halts execution immediately

  For example, the SX-1748 message indicates that the schematic top block name is not specified. The extraction cannot proceed.

Computing Environment Warning Messages

If the StarRC tool detects computing environment issues, the tool issues an SX-3564 warning message, which is included in the standard summary file. In addition, a file named sml.sum is created that contains detailed SML warning messages for the conditions shown in Table 2. For more information, see the warning message man pages.

Table 2 Warning Messages Related to the Computing Environment

Condition	Message ID	Description
Low memory availability	SML-001	System memory usage is over 98 percent
High CPU usage	SML-002	Host CPU load is over 95 percent, although the Synopsys tool process is using only 5 percent or less of the CPU capacity
Network congestion issue	SML-003	Packet retransmission rate is over 5 percent, indicating network congestion
Network connectivity issue	SML-004	Packet loss is over 5 percent, indicating poor network connectivity
High network latency	SML-005	Latency is over 300 ms from the host running the process to hosts specified in the message

Feedback

StarRC Message Man Pages

Man pages for formal messages provide information that supplements the message that you see in your log file or interactive session. Each man page contains the text of the actual message, a description of the condition, and some suggestions for followup actions.

Use the `starrc_man` command at the operating system prompt to view a man page. For example,

```
% starrc_man SX-1748
```

The `starrc_man` command is installed in the same directory that contains the StarXtract executable. If you install the StarRC tool using the standard installation procedure, which includes setting the PATH environment variable, the `starrc_man` command is also installed.

Error Message Control

You can control how error messages are issued with the following StarRC commands:

- The `CONVERT_WARNING_TO_ERROR` command allows you to halt extraction when the tool encounters specified warning conditions.
- The `MESSAGE_SUPPRESSION` command limits the number of times that specific messages are issued. The limit applies only to messages whose IDs are listed in the command.
- The `MESSAGE_SUPPRESSION_LIMIT` command applies a limit to all warning, error, and information messages.

These features apply only to SX, EX, and GRD messages, which are messages issued by the StarRC tool. During an extraction run, messages with different prefixes might appear.

In the following example, the StarRC tool stops reporting most messages after the tenth occurrence, but reports EX-269 messages up to twenty times:

```
MESSAGE_SUPPRESSION_LIMIT: 10
MESSAGE_SUPPRESSION: EX-269:20
```

2

Running StarRC

This chapter provides basic information about the tools provided in the StarRC installation.

For more information, see the following topics:

- StarRC Overview
- The grdgenxo Tool and nxtgrd Files
- The oasis_info Utility
- The StarXtract Command
- Distributed Processing

Feedback

StarRC Overview

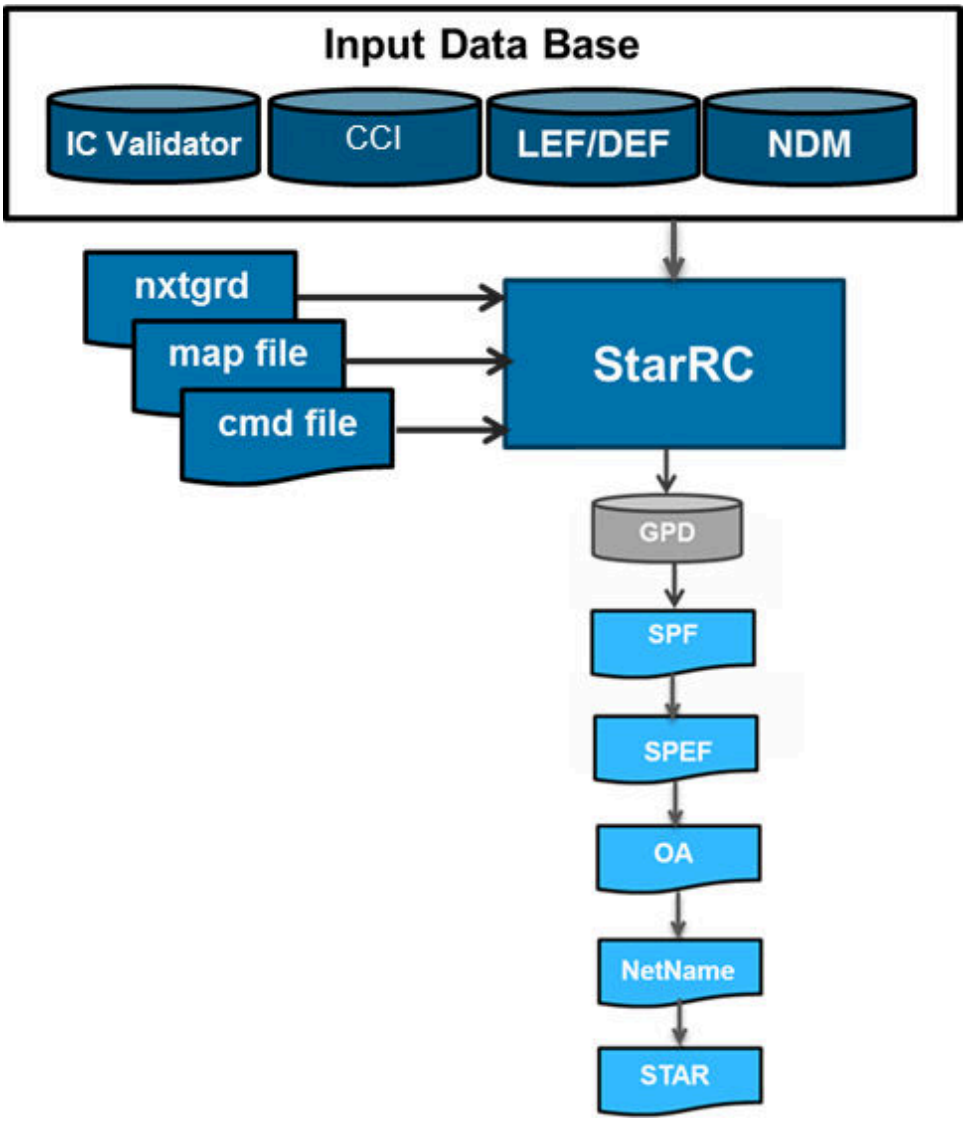
The StarRC tool analyzes the parasitic capacitances and resistances of advanced circuit designs. The tool uses pattern matching to analyze a design by comparing the layout to a set of primitive structures whose parasitics have been previously simulated. The reference parasitics are contained in a process characterization database called the nxtgrd file.

- For gate-level extraction, the StarRC tool accepts LEF/DEF designs and design libraries created by the Fusion Compiler or IC Compiler II tool in the New Data Model (NDM) format.
- For transistor-level extraction, the StarRC tool reads output from the IC Validator, Hercules, and Mentor Graphics® Calibre® layout-versus-schematic (LVS) tools.
- The StarRC tool uses foundry-created nxtgrd files to represent the effects of the manufacturing process.
- The tool saves extracted parasitics in the binary parasitics database (GPD). You can optionally create netlist files in other formats such as SPF and SPEF.

Figure 3 illustrates the general StarRC flow.

Feedback

Figure 3 The StarRC Flow



You can use the StarRC tool in different ways to accomplish different goals. The primary StarRC flows are as follows:

- Gate-level (cell-level) extraction

  In a gate-level flow, the design is composed of cells (macros). Gate-level extraction is commonly used to analyze the entire chip for timing analysis in a signoff loop. By default, cells are treated as skipped cells (or skip cells), which means that the capacitance of nets inside the cells is not analyzed.

- Transistor-level extraction

The transistor-level flow focuses on smaller portions of the design in greater detail. You can examine parasitics at the device level. Information from a layout-versus-schematic (LVS) tool is required for this flow.

- Special-purpose flows
  - Clock net inductance analysis
  - Field solver analysis

Some StarRC features or commands are restricted to a specific flow, as stated in the command reference pages or in feature descriptions within this user guide.

The grdgenxo Tool and nxtgrd Files

An nxtgrd file contains the reference parasitics for the manufacturing process of the chip. Using an nxtgrd file obtained from a foundry is the most accurate way of analyzing the parasitics of a specific process technology. The nxtgrd file is specified in the StarRC command file with the `TCAD_GRD_FILE` command.

For process exploration, you can change some of the process parameters and create your own nxtgrd files. However, the results might not correlate with manufactured devices unless you use an nxtgrd file created specifically by the foundry for a specific manufacturing process.

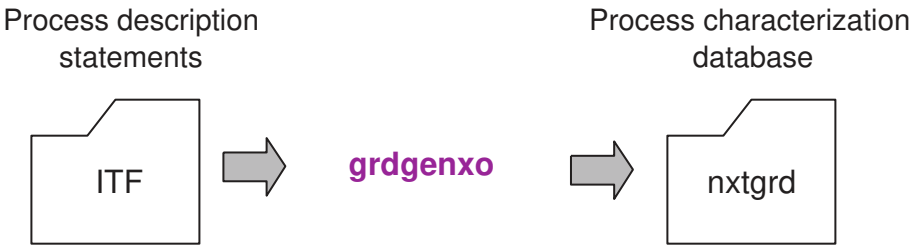
An nxtgrd file is created by the grdgenxo tool, a StarRC utility program, as illustrated in Figure 4. The grdgenxo tool generates the nxtgrd file from a file written in a process description language called the Interconnect Technology Format (ITF). The grdgenxo tool uses an internal field solver operating on an extensive set of primitive structures. The nxtgrd file contains capacitance, resistance, and layer information, along with the ITF statements.

You can use ITF files in two ways:

- Run the grdgenxo tool to process the ITF file and create an nxtgrd file for use in the StarRC tool (specified with the `TCAD_GRD_FILE` command).
- Use the StarRC tool to read the ITF file directly by using the `ITF_FILE` command. The tool automatically uses the field solver to analyze the capacitance of the design. This operation is supported only for capacitance extraction in transistor-level flows.

Feedback

Figure 4 Process Database File Generation



For reference information about the ITF process description statements, see *ITF Statements*.

For more information about the grdgenxo tool and process modeling, see *Process Modeling Methodology* and *Process Characterization*.

The oasis_info Utility

The StarRC installation includes the oasis_info utility for checking the integrity of OASIS files. Execute this utility from the operating system prompt, as follows:

```
% oasis_info
```

The oasis_info utility provides a report to stdout unless you redirect it to a log file. Error, warning, and information messages are generated outside of the StarRC tool and therefore do not have StarRC message IDs or man pages. An example of the information message is as follows:

```
****************************************
* INFO: Found no CELL record for CELLNAME (name:ram8x64) record.
```

An example of the report is as follows:

```
Record type        Counts  Array_Rep(Instances)      Random_Rep(Instances)
-------------------------------------------------------------------------
1  CELLNAME:3       598         0         0                0          0
2  TEXTSTRING:5    1634         0         0                0          0
...
7  PLACEMENT:17    1343      1073      5171              224     115371
8  TEXT:19         6408      1082      2164                0          0
...
-------------------------------------------------------------------------
   Total counts   37758      4505     13353             4685     224087

CBlocks Records:  526       35781( 94.8%) records in CBlocks.
Compressed 1455035 bytes to 1151448 bytes (79.1%)

CELLNAME:  strict mode with S_CELL_OFFSET property records.
```

```
TEXTSTRING: strict mode
...

/..../folder1/folder2/top.oas is an OASIS compliant file.
```

You must have the STAR-RC2_MANAGER and STAR-RC2_ULTRA_MANAGER licenses to use the oasis_info utility. For best results, save OASIS files in strict mode for memory efficiency and performance and ensure that your files are OASIS compliant.

The StarXtract Command

Invoke the StarRC tool by using the `StarXtract` command at the operating system prompt. Table 3 provides brief descriptions of the command options and links to more information. The general command syntax is as follows:

```
% StarXtract option1 option2 ... cmd_file1  [cmd_file2] ...
```

Table 3 Command-Line Options for the StarXtract Command

Option	Description
cmd_file1	A StarRC command file; at least one is required.
cmd_file2, ...	Optional additional command files.
-cleanN	Regenerates the output netlist; for transistor-level extraction only. For more information, see The StarXtract -cleanN and -clean_converter Options.
-clean_converter	Regenerates an output netlist; for transistor-level GPDs only. For more information, see The StarXtract -cleanN and -clean_converter Options.
-compare_parasitics	Compares two netlists. For more information, see The StarXtract -compare_parasitics Option.
-Display	Converts opens and shorts data for selected nets to a database format for display. For more information, see Display Options for Debugging Opens and Shorts.
-Display_mf	Converts metal fill shorts data for selected nets to a database format for display. For more information, see Display Options for Debugging Opens and Shorts.
-Display remove_unused	Converts shorts data for selected nets to a database format for display. For more information, see Display Options for Debugging Opens and Shorts.
-Display short_regions	Converts shorts data for selected nets to a database format for display. For more information, see Display Options for Debugging Opens and Shorts.

Feedback

Table 3 Command-Line Options for the StarXtract Command (Continued)

Option	Description
`-write_short_regions`	Saves information in the vicinity of shorts for later use with the Parasitic Explorer tool. For more information, see The StarXtract -write_short_regions Option.
`-gdscheck`	Parses a GDSII file and checks for errors. For more information, see The StarXtract -gdscheck Option.
`-convert_gpd_to_spef`	Creates a SPEF netlist from a GPD. For more information about GPD conversion options, see The Parasitic Database or GPD.
`-convert_gpd_to_spf`	Creates an SPF netlist from a GPD. For more information about GPD conversion options, see The Parasitic Database or GPD.
`-convert_gpd_to_oa`	Creates an OA netlist from a GPD for transistor-level flows. For more information about GPD conversion options, see The Parasitic Database or GPD.
`-dump_gpd_config`	Generates an ASCII configuration file for a GPD.
`-set_gpd_config`	Applies a configuration file to a GPD.
`-reset_gpd`	Resets a GPD to its original state.
`-merge_corner_parasitics`	Creates a new GPD from different corners of an existing GPD. For more information, see The StarXtract -merge_corner_parasitics Option.
`-ultra`	Uses only the STAR-RC2_ULTRA_MANAGER license key. For more information, see StarRC Licensing.
`-cdnlicsvr`	Runs the Virtuoso® Integration license server.
`-tech_out`	Displays a list of StarRC command options and their default, if applicable. If you specify this option in a StarRC command file, the output of the StarRC command file displays • The specified options and their corresponding settings • The default of the options that are not specified
`-v`	Displays the StarRC version.
`-h`	Displays the command usage report.
`-iapinetmap`	Displays the net name or id mapping for IAPI.
`-iapixindump`	Displays the ASCII xin output for IAPI.
`-pio`	Writes the PIO file from Milkyway.

Feedback

Table 3 Command-Line Options for the StarXtract Command (Continued)

Option	Description
`-skip`	Writes the skip cells from Milkyway models.

The StarRC Command File

A StarRC command file is a list of commands that specify conditions for the extraction run. You can specify multiple command files with the `StarXtract` command. For example, you might want to include general setup commands in one command file and design-specific commands in a separate file.

Commands in separate command files are treated as if they were written in a single command file, in the order provided. The first command file is read first, followed by the second command file, and so on. If StarRC commands are duplicated, later instances of a command usually overwrite earlier instances. However, some commands that contain lists of objects are cumulative.

You can also use the `INCLUDE_FILE` command to specify the name of a command file, in which case the commands are inserted at the location of the `INCLUDE_FILE` command.

For information about how the StarRC tool treats multiple instances of specific commands, see the reference pages in Chapter 14, StarRC Commands.

A line that begins with an asterisk (*) is a comment. Command names are not case-sensitive. The terms statement, command, option, and keyword are synonymous.

Feedback

The StarXtract -cleanN and -clean_converter Options

The `-cleanN` option is valid only for transistor-level flows in which a GPD is not created due to the use of an unsupported command. For GPD flows, use the `-clean_converter` option.

The -cleanN Option

You can create a new netlist from a completed transistor-level extraction run by using the `StarXtract -cleanN` command. If the initial run generates a netlist in the `SPF`, `OA`, or `NETNAME` formats, you can create a new netlist in any format. However, if the initial run generates a SPEF netlist or does not generate any netlist, you can create only SPEF netlists with the `StarXtract -cleanN` command.

The `-cleanN` option is valid only for transistor-level flows that do not create a GPD. The following commands are allowed:

```
COUPLE_NONCRITICAL_NETS_PREFIX
COUPLE_NONCRITICAL_NETS_SUBNODE_SUFFIX
COUPLING_REPORT_FILE
COUPLING_REPORT_NUMBER
LPE_DEVICES
LPE_PARAM
NETLIST_* commands
NONCRITICAL_COUPLING_REPORT_FILE
OA_* commands
PRINT_SILICON_INFO
REMOVE_NET_PROPERTY
SHEET_COUPLE_TO_NET
SHEET_COUPLE_TO_NET_LEVEL
SLEEP_TIME_AFTER_FINISH
SPICE_SUBCKT_FILE
```

The -clean_converter Option

The `-clean_converter` option is valid only for transistor-level GPD flows. The following commands are allowed:

```
NETLIST_COMMENTED_PARAMS
NETLIST_COMPRESS_COMMAND
NETLIST_CONNECT_SECTION
NETLIST_COUPLE_UNSELECTED_NETS
NETLIST_DEVICE_LOCATION_ORIENTATION
NETLIST_FILE
NETLIST_FORMAT
NETLIST_NODENAME_NETNAME
NETLIST_PRINT_CC_TWICE
NETLIST_GROUND_NODE_NAME
NELIST_INSTANCE_SECTION
NETLIST_NODE_SECTION
NETLIST_PASSIVE_PARAMS
```

```
NETLIST_SELECT_NETS
NETLIST_SUBCKT
```

The `NETLIST_COMMENTED_PARAMS` and `NETLIST_INSTANCE_SECTION` commands are valid only if they were set to `YES` in the initial extraction.

Feedback

The StarXtract -compare_parasitics Option

Use the `-compare_parasitics` option of the `StarXtract` command to compare two sets of parasitic data and generate reports about the differences. You can compare the entire set of data or restrict the comparison to a specific set of paths or nets.

The two sets of parasitic data must have the same format. Supported formats are as follows:

- Data in the parasitic database (GPD)
- SPEF, SPF, and DSPF netlists in either compressed or uncompressed form

The command syntax is as follows. Table 4 describes the options and their arguments.

```
StarXtract -compare_parasitics test  reference
           [-cores  num_cores]
           [-tcap  lumpC_abs]
           [-ccap  CC_abs   CC_rel]
           [-res   P2P_abs]
           [-r] [-c] [-d]
           [-corner "c_1"]
           [-match name | xy]
           [-net net_name]
           [-from_pin start_pin_name | start_pin_xy]
           [-to_pin end_pin_name| end_pin_xy]
           [-net_config cfile_name]
           [-timeout time_limit]
           [-delay_pin_load]
           [-delay delay_abs]
           [-solver PardisoX]
```

Note:

These options of the `StarXtract` command are valid only when used with the `-compare_parasitics` option.

Table 4 Options and Arguments for Parasitics Comparison

Option	Argument	Default	Description
`-cores`	*num_cores*	1	Number of cores for distributed processing Type: integer
`-compare_parasitics`	*test* *reference*	none none	Data set to compare to the reference data Reference data set Type: Netlist file name or GPD directory name
`-tcap`	*lumpC_abs*	3 fF	Compares only the nets in the reference data that have total capacitance equal to or larger than *lumpC_abs*

Feedback

Table 4 Options and Arguments for Parasitics Comparison (Continued)

Option	Argument	Default	Description
`-ccap`	*CC_abs* *CC_rel*	0.3 fF 0.1	Compares only the nets in the reference data that have absolute coupling capacitance equal to or larger than *CC_abs* and relative coupling capacitance equal to or larger than *CC_rel*. Both conditions must be met. Relative coupling capacitance is the ratio of absolute coupling capacitance (*CC_abs*) to total capacitance (*lumpC_abs*).
`-res`	*P2P_abs*	50 ohms	Compares only the nets in the reference data that have point-to-point resistance equal to or larger than *P2P_abs*
`-r`	none	none	Performs resistance comparison
`-c`	none	none	Performs capacitance comparison
`-d`	none	none	Performs Elmore delay comparison
`-corner`	*c_1*	none	Performs comparison for the specified corner from a simultaneous multicorner extraction run. Valid only when comparing GPD data
`-match`	`name` or `xy`	`name`	Specifies pin matching mode
`-net`	*net_name*	none	Specifies a single net name. Calculates the total capacitance, all coupling capacitances, and all point-to-point resistances for the specified net. Cannot be used with the `-from_pin` or `-to_pin` options.
`-from_pin`	*start_pin_name* OR *start_pin_xy*	none	Calculates the point-to-point resistances for all paths that originate from the pin. If you use both the `-from_pin` and `-to_pin` options, calculates resistance for paths between the two pins. Use pin names for name-based matching using the `-match name` option. Use pin coordinates for location-based matching using the `-match xy` option. Provide coordinates in microns separated by a comma and no spaces.

Feedback

Table 4 Options and Arguments for Parasitics Comparison (Continued)

Option	Argument	Default	Description
`-to_pin`	*end_pin_name* OR *end_pin_xy*	none	Calculates the point-to-point resistances for all paths that originate from the pin. If you use both the `-from_pin` and `-to_pin` options, calculates resistance for paths between the two pins. Use pin names for name-based matching using the `-match name` option. Use pin coordinates for location-based matching using the `-match xy` option. Provide coordinates in microns separated by a comma and no spaces.
`-net_config`	*cfile_name*	none	Specifies the name of a net configuration file, which can contain multiple net and path specifications.
`-timeout`	time_limit	5400 s	Limits the calculation time per net
`-delay_pin_load`	none	NO	Includes the pin capacitance at the sink node in the delay calculation
`-delay`	delay_abs	1 ps	Specifies the minimum delay for consideration, in ps
`-solver`	`PardisoX`	none	Use the PardisoX (Parallel Direct Sparse) solver interface to improve point-to-point comparison runtime

In the following example, netlist file new.spef.gz is compared to netlist file old.spef.gz. Only those nets that have total capacitance of 3 fF or more, absolute coupling capacitance of 0.3 fF or more, relative coupling capacitance ratio of 0.1 or more, and point-to-point resistance greater than 50 ohms are included in the comparison. Other nets are ignored.

```
% StarXtract -compare_parasitics new.spef.gz old.spef.gz -tcap 3    \
           -ccap 0.3 0.1 -res 50
```

The following usage notes apply to GPD comparisons:

- To specify GPD parasitics, provide the name of the GPD directory. An example is *ref.gpd*.
- If you compare two GPDs that were both generated with simultaneous multicorner (SMC) extraction, you must use the `-corner` option. The specified corner name must exist in both of the GPDs.

Feedback

- If you compare a GPD that was generated with SMC extraction to a GPD that did not use SMC, you must use the `-corner` option to name a corner in the GPD that used SMC.
- If you compare two GPDs that were generated without SMC extraction, do not use the `-corner` option.

Comparing Specific Paths

You can compare specific paths by identifying startpoint pins, endpoint pins, or both. You can identify pins by name or by xy location. In addition, you can provide a text file that contains a list of paths for the comparison.

The behavior of the pin identification options is as follows. Note that the `-r`, `-c`, and `-d` options affect the specific parasitics that are compared. The maximum number of pin combinations for a single net is 250.

- If you specify a single net with the `-net` option, the tool calculates the total capacitance, all coupling capacitances, all point-to-point resistances, and the Elmore delay for the specified net. This option cannot be used with the `-from_pin` and `-to_pin` options.
- If you specify a startpoint with the -`from_pin` option, the tool calculates the point-to-point resistance for each path that originates at that pin.
- If you specify an endpoint with the `-to_pin` option, the tool calculates the point-to-point resistance for each path that ends at that pin.
- If you specify both the `-from_pin` and `-to_pin` options, the tool calculates the point-to-point resistance between the two pins. Both pins must be on the same net.

For point-to-point capacitance and resistance comparisons, the following pin combinations are supported:

- Gate-level GPD or SPEF file comparisons
  - From an input or bidirectional pin to an output or bidirectional pin
  - From an output or bidirectional pin to an input or bidirectional pin
  - The from and to pins must not be the same type
- Transistor-level GPD or SPF file comparisons
  - From an input, output, or bidirectional pin to an input, output, or bidirectional pin

Feedback

Name-Based Comparison With the -match name Option

If you use the `-match name` option, the compare parasitics utility uses pin names to match the starting and ending pins between the two databases. This is the default behavior if you do not use the `-match` option.

You can specify a single net name for comparison by using the `-net` option. You can also use the `-from_pin` and `-to_pin` options with pin names to specify path startpoints, endpoints, or both.

To compare a set of specific nets, you can also provide them in a net configuration file specified with the `-net_config` option. For name-based matching, use pin names with the `FROM_PIN`, `TO_PIN`, and `FROM_TO_PINS` statements in the net configuration file. You can also specify net names with the `NET` statement.

For each net, the tool calculates the total capacitance, all coupling capacitances, all point-to-point resistances, and the Elmore delay for the specified net. Output reports contain pin names.

In the following example, the parasitics on the net named test_net are compared between netlist files spef1 and spef2.

```
% StarXtract -compare_parasitics spef1 spef2 -net "test_net"
```

In the following example, the parasitics on all nets originating from the pin named clk_out are compared between netlist files spef1 and spef2.

```
% StarXtract -compare_parasitics spef1 spef2 -match name \
             -from_pin "clk_out"
```

Using a net configuration file makes performing multiple comparisons simpler and repeatable. A net configuration file for the previous examples contains entries as follows:

```
** Nets to compare for design ABC
NET: test_net
FROM_PIN: clk_out
```

The following command compares the parasitics on nets in the net configuration file (named comp_nets), using name-based pin matching:

```
% StarXtract -compare_parasitics spef1 spef2 -match name \
             -net_config comp_nets
```

Location-Based Comparison With the -match xy Option

If you compare two data sets that were generated with different LVS flows, some pin names might not match due to swapping of equivalent nets or devices during cross-referencing operations. In this case, you can use the `-match xy` option to perform the comparison using pin locations instead of pin names.

Feedback

You can use the `-from_pin` and `-to_pin` options with pin xy coordinates (in microns) to specify path startpoints, endpoints, or both. You must provide the xy coordinates as a pair of values separated by a comma and no spaces.

To compare a set of specific nets, you can also provide them in a net configuration file specified with the `-net_config` option. For location-based matching, use pin xy coordinates with the `FROM_PIN`, `TO_PIN`, and `FROM_TO_PINS` statements in the net configuration file.

For each net, the tool calculates the total capacitance, all coupling capacitances, all point-to-point resistances, and the Elmore delay for the specified net. Output reports contain pin coordinates.

In the following example, the parasitics on all nets originating from the pin located at xy coordinates (431.6,558) are compared between netlist files spef1 and spef2.

```
% StarXtract -compare_parasitics spef1 spef2 -match xy \
              -from_pin "431.6,558"
```

In the following example, the parasitics on all nets between the pin located at xy coordinates (431.6,558) and the pin located at (285.6,590.5) are compared between netlist files spef1 and spef2:

```
% StarXtract -compare_parasitics spef1 spef2 -match xy \
              -from_pin "431.6,558" -to_pin "285.6,590.5"
```

Using a net configuration file makes performing multiple comparisons simpler and more repeatable. A net configuration file for the previous examples is as follows:

```
** Nets to compare for design XYZ
FROM_PIN: 431.6,558
FROM_TO_PINS: 431.6,558 285.6,590.5
```

The following command compares the parasitics on nets in the net configuration file named comp_nets:

```
% StarXtract -compare_parasitics spef1 spef2 -match xy \
              -net_config comp_nets
```

The Net Configuration File

Table 5 describes the statements that are valid in a net configuration file.

Table 5 Net Configuration File Statements

Statement and Arguments	Description
//	Begin a comment line
**	Begin a comment line

Feedback

Table 5 Net Configuration File Statements (Continued)

Statement and Arguments	Description
NET: *net_name*	Specify a single net name. Valid with both the `-match name` and `-match xy` options.
FROM_PIN: *start_pin*	Specify a single pin as a path startpoint or source pin. Use pin names for name-based matching using the `-match name` option. Use pin coordinates for location-based matching using the `-match xy` option. Provide coordinates in microns separated by a comma and no spaces.
TO_PIN: *end_pin*	Specify a single pin as a path endpoint or sink pin. Use pin names for name-based matching using the `-match name` option. Use pin coordinates for location-based matching using the `-match xy` option. Provide coordinates in microns separated by a comma and no spaces
FROM_TO_PINS: *start_pin end_pin*	Specify path startpoint and endpoint pins (source and sink pair). Use pin names for name-based matching using the `-match name` option. Use pin coordinates for location-based matching using the `-match xy` option. Provide coordinates in microns separated by a comma and no spaces

Comparing the Elmore Delay

Elmore delay is an approximation to the RC delay of a net. For a specific pair of pins, the signal direction can affect the delay. Therefore the report always includes the delay in both directions.

The `-delay` option specifies the minimum delay for which to make the comparison. For a specific path, the delay in both the reference data set and the test data set must be larger than or equal to the specified minimum delay to be included in the report.

For Elmore delay comparisons, the following pin combinations are supported:

- From a top-level (*P) input or bidirectional pin to a top-level output or bidirectional pin
- From a top-level input or bidirectional pin to an instance-level input or bidirectional pin
- From an instance-level (*I) output or bidirectional pin to a top-level input or bidirectional pin
- From an instance-level output or bidirectional pin to an instance-level input or bidirectional pin

Top-level output pins and instance-level input pins cannot serve as source pins. Similarly, top-level input pins and instance-level output pins cannot serve as sink pins.

In the following example, a comparison of total capacitance, coupling capacitance, and Elmore delay is performed between netlist files spef1 and spef2 for all nets. The Elmore delay considers only nets with a delay of 2 ps or more and includes the output pin capacitance in the analysis.

```
% StarXtract -compare_parasitics spef1 spef2 -tcap 3 -ccap 0.3 0.1 \
              -res 50 -d -delay 2 -delay_pin_load
```

Output Files

Table 6 lists the output files generated by the compare parasitics utility.

Table 6 Parasitics Comparison Output Files

File name	Contents
tcap.rpt	Total capacitance differences between reference and tested nets
ccap.rpt	Coupling capacitance differences between reference and tested nets
p2p.rpt	Point-to-point resistance differences between reference and tested nets
nets.mismatched	Nets that exist in only one of the netlists
delay.rpt	Elmore delay report
summary.rpt	Summary of results

Feedback

Display Options for Debugging Opens and Shorts

The `StarXtract` command used with one of the display options does not perform extraction. Instead, the `-Display`, `-Display_mf`, and `-Display short_regions` options provide the capability to debug opens and shorts identified in a previous StarRC extraction run. These options use information saved in the star directory to create a special-purpose database that contains selected signal nets along with nearby polygons associated with opens and shorts on those nets. You can then use a layout viewing tool, such as the Fusion Compiler and IC Compiler II tools, to view the generated database.

Use the following `StarXtract` command options to save data for debugging:

- The `-Display` option saves opens on critical nets (signal nets) and shorts between critical nets.
- The `-Display_mf` option saves shorts between critical nets and metal fill polygons.
- The `-Display_mf close_net` option identifies the bounding box for each selected net and scales the size of the bounding box area by multiplying the size with the scale factor. The bounding box of a net is defined as the minimal rectangle area that wraps the entire net. The scaled area is known as a configurable area, where

  ```
  configurable area = bounding box area * scale factor
  ```

- The `-Display short_regions` option saves shorts between critical nets and noncritical polygons such as blockages, skip cell internal polygons, unextracted power nets, and unselected signal nets.
- The `-Display remove_unused` option removes unused layers to allow use of a Milkyway format design database that has more than 256 layers.

This feature is available for gate-level flows that do not use the field solver. Nets are displayed in mask layout dimensions after the application of any half-node scale factors.

The display options require a simple StarRC command file to generate the debugging database. The command file must contain one of the `DEBUG_NDM_DATABASE` or `DEBUG_MILKYWAY_DATABASE` commands to specify the name of the debugging database. The debugging database format is independent of the format of the original design.

To investigate opens and shorts, follow this procedure:

1. Perform a StarRC extraction run.
2. Create a simple StarRC command file for the purpose of visualizing opens and shorts. In the following example, the `NETS` command selects specific nets to view; selecting all nets is not recommended, to save runtime.

   ```
   DEBUG_NDM_DATABASE: my_design
   STAR_DIRECTORY: star
   NETS: net1 net2 net3
   ```

Feedback

Use either the DEBUG_NDM_DATABASE or DEBUG_MILKYWAY_DATABASE command, depending on the database format that you prefer for use in a layout viewing tool.

3. (Optional) If you plan to examine shorts between critical nets and noncritical polygons with the -Display short_regions option, include the following command in the command file:

```
ENHANCED_SHORT_REPORTING: COMPLETE
```

4. Invoke the StarRC tool with one of the display options and the name of the debugging command file. For example:

```
% StarXtract -Display short_regions star_cmd_debug
```

The StarRC tool saves a region of the design expanded around the site of each detected short.

Note:

If you are saving the data into a Milkyway format database and the design contains more than 256 layers, use the remove_unused option to reduce the number of layers in the design. Otherwise, some layers might not be visible in the layout viewer. The remove_unused option must appear at the end of the command, as follows:

```
% StarXtract -Display short_regions star_cmd_debug
 remove_unused
```

5. Open a layout viewer to examine the new database. Figure 5 illustrates the saved region for the -Display short_regions option.

Figure 5 Region Saved Around a Short

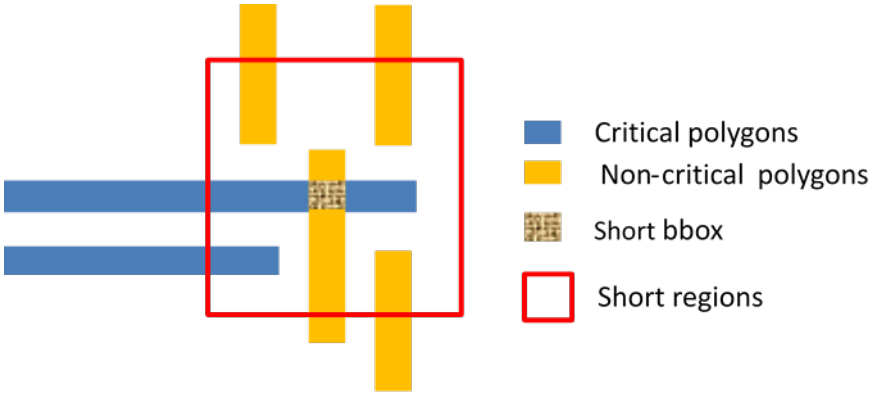


Figure 6 is an example of a net identified as an open by the StarRC tool. Examination reveals that a via is missing between the M1 and M2 layers.

Feedback

Figure 6 View of Signal Net Open

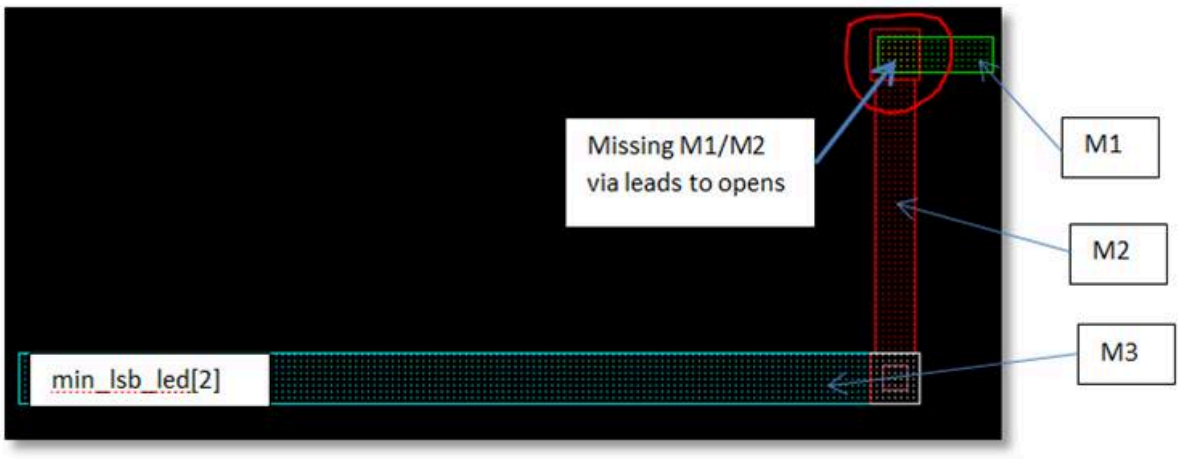
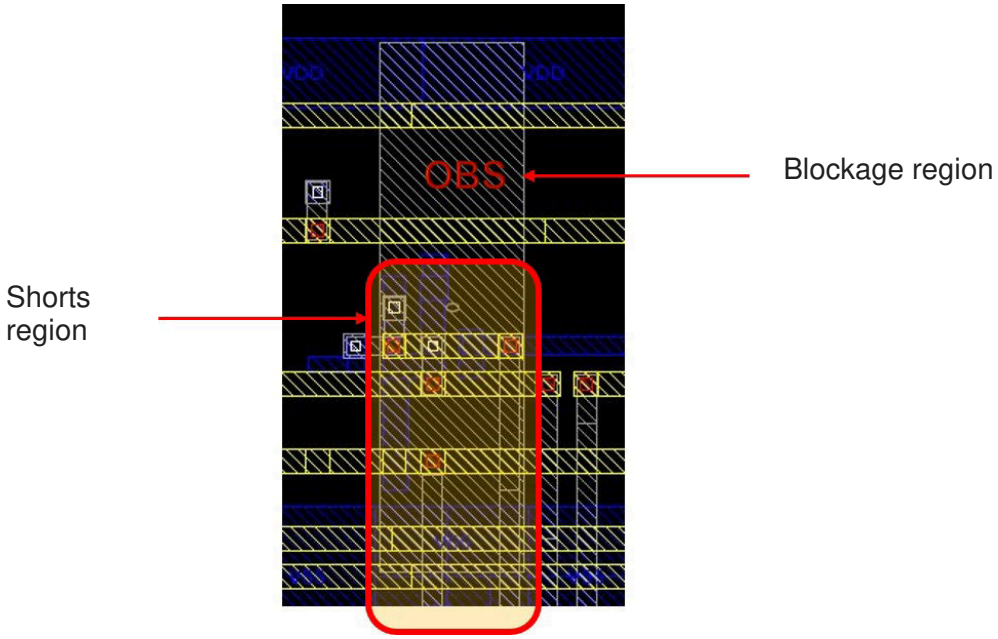


Figure 7 is an example of a region where signal nets are shorted to a blockage region.

Figure 7 Example View of Noncritical Short Polygons



The StarXtract -gdscheck Option

The `-gdscheck` option parses a specified GDSII file and checks it for problems. Table 7 lists the checks and the error reporting methods.

Feedback

Table 7 GDSII File Checks

Check Type	Description	Error Reporting
Record length	Checks that record length is 4 to 65535 bytes, inclusive	SX-0202 message
Record data type	Checks that record data type is 0 to 6, inclusive Checks that every record has a data type	SX-0205 and SX-0206 messages
Consecutively duplicate elements	Removes consecutively duplicate elements	Reports duplicate elements in the summary file SX-1701 message if more than 4 duplicate elements

For example, the following command checks the contents of file ABC.gdsii:

```
StarXtract -gdscheck ABC.gdsii
```

The output is written to standard output (stdout) and, if not redirected, appears only on the terminal session. The results for each cell appear between BEGIN_CELL and END_CELL statements. An example of the output is as follows:

```
Parsing GDS file ABC.gdsii
BEGIN_CELL MM20_ADD_1
END_CELL
BEGIN_CELL MM56_B
END_CELL
BEGIN_CELL RDEL23
END_CELL
WARNING: More than 4 consecutively duplicate elements found in the GDS
 file, it might incur extraction problems (SX-1701)
Consecutively duplicate GDS elements
 POLY: 4
 PATH: 0
```

Feedback

The StarXtract -merge_corner_parasitics Option

A single GPD (the binary parasitic database) might contain parasitics from up to 15 process corners obtained during simultaneous multicorner extraction. The `StarXtract -merge_corner_parasitics` command allows you to mix and match parasitics from different corners and save them into a new GPD. You can then create a SPEF or SPF output file from the merged GPD for use in simulation tools.

For example, you might want to use the parasitics from corner A for data paths and the parasitics from corner B for clock paths.

This feature requires the Ultra+ license.

The command syntax is as follows. Table 8 describes the options and their arguments.

```
StarXtract -merge_corner_parasitics source_GPD  dest_GPD
           -base_corner  base_corner
           [-select_corner  corner1 file1 corner2 file2 ... ]
```

Table 8 Options and Arguments for Corner Merging

Option	Argument	Description
`-merge_corner_parasitics`	*source_GPD* *dest_GPD*	Source GPD containing 2 or more corners Destination GPD; must not already exist Type: GPD directory name
`-base_corner`	*base_corner*	Default corner to use for all nets not specified in a `-select_corner` option argument
`-select_corner`	*cornerN* *fileN*	A corner and a file that lists nets whose parasitics should come from that corner. The argument list can contain more than one corner-file pair. A specific net must not appear in more than one of these files.

To obtain the names of the corners saved in the source GPD, use one of the following methods:

- Examine the README file in the top-level directory of the source GPD.
- In the Parasitic Explorer tool, use the `get_gpd_corners` command. For more information, see the *Parasitic Explorer User Guide*.

The percentage of nets that are named in select corners files is expected to be relatively small. Specifying a large number of nets in these files results in performance degradation.

Coupling symmetry is maintained by including the select net coupling on both victim and aggressor nets.

Feedback

The select nets file must follow these guidelines:

- Net names must be specified in fully expanded form.
- Wildcards *, ?, and ! are acceptable.
- A net must not appear in more than one file.
- Parasitics for nets that are not named in any file come from the base corner.

The StarRC tool saves a list of matched nets and their associated corners in a file named README.selected_nets, which is located in the top-level directory of the destination GPD. Each line in the file contains information about a single net in the following format:

```
corner_name   net_name
```

If you generate a SPEF output file from a GPD that uses merged corners, the comments section lists the corners referenced in the file in the following format:

```
// CORNER_NAME merged_cornerB1:cornerM1:cornerM2: ... cornerMn
```

In this example, cornerB1 is the base corner and the other corners (cornerM1 and so on) are the merged corners.

For example:

```
// CORNER_NAME merged_typical:temp125:fast
```

The StarXtract -write_short_regions Option

The `-write_short_regions` option saves layout information in the vicinity of shorts for later use with the Parasitic Explorer tool.

Although the StarRC tool finds many shorts during extraction, the tool cannot save detailed information about the design environment around every short. If you plan to use the Parasitic Explorer tool to query the contents of the GPD, you can ensure that the extra information is saved for specified nets or regions by using the `-write_short_regions` option with the `StarXtract` command when you perform the extraction.

The command syntax is as follows. Table 9 describes the options and their arguments.

```
StarXtract -write_short_regions \
           -nets_file nets | -window llx lly urx ury \
           cmd_file
```

Feedback

Table 9 Options and Arguments for Saving Shorts Data

Option	Argument	Description
`-write_short_regions`	*cmd_file*	StarRC command file for the extraction
`-nets_file`	*nets*	A file containing a list of nets
`-window`	*llx lly urx ury*	Coordinates of the region of interest (lower-left x-coordinate, lower-left y-coordinate, upper-right x-coordinate, upper-right y-coordinate). The coordinate unit is micron.

The saved information is used when you execute the `starrc_gpd_read_opens_shorts` command in the Parasitic Explorer tool. For more information, see the *Parasitic Explorer User Guide*.

Feedback

Distributed Processing

Using distributed processing can greatly reduce the elapsed time for complex runs by distributing the computational load during all phases of StarRC runs.

The StarRC tool ensures consistency of parasitic results, independent of the number of cores involved.

To specify distributed processing, set the `NUM_CORES` command in the StarRC command file to a value greater than 1 and use one of the following methods to start the runs:

- Manual submission: Start multiple runs in the same directory, using the same StarRC command file for each run
- Automatic submission: Set up job submission details for your computing environment by using the `STARRC_DP_STRING` command or the `STARRC_DP_STRING` environment variable, then start one master StarRC run.

Note:

The number of worker processes launched by the StarRC tool is equal to the setting of the `NUM_CORES` command. If your submission command specifies a larger number of cores, some cores are reserved but not used. For best results, the StarRC `NUM_CORES` command and the submission command should specify the same number of cores.

The star directory (the directory specified by the `STAR_DIRECTORY` command) must be accessible by all StarRC processes, whether launched manually or automatically. If all processes are run on a single host, use a disk local to that host for optimal performance. If processes are to be distributed across multiple hosts, the star directory must be located on a network disk that is accessible from all of the hosts.

The available StarRC licenses might limit the number of processes that can run in parallel.

For site-specific information about job submission commands and guidelines, contact your system administrator.

The following topics provide detailed information about distributed processing:

- Manual Submission of Distributed Processing Runs
- Automatic Submission of Distributed Processing Jobs
- Distributed Processing Error and Message Handling
- Distributed Processing Reports

Feedback

Manual Submission of Distributed Processing Runs

With manual submission, you begin a StarRC run on a single host and then submit runs to additional hosts, using the protocols for your computing environment. You must use the same StarRC command file for all runs. You must execute the `StarXtract` command from the same directory for all hosts.

Concurrent execution is limited to the number of runs specified by the `NUM_CORES` setting in the StarRC command file. Additional submitted runs terminate with an error message.

Do not use the `STARRC_DP_STRING` command or the `STARRC_DP_STRING` environment variable with manual submission.

Automatic Submission of Distributed Processing Jobs

With automated submission, you start a single run and let the StarRC tool automatically submit multiple jobs according to the computing environment protocol specified in the `STARRC_DP_STRING` command or the `STARRC_DP_STRING` environment variable. You can also use this method to run a single job remotely.

The information in this section does not apply to distributed processing runs invoked by a GPD configuration file or by the field solver (whose runs are controlled by the `FS_DP_STRING` command).

Job submission commands are site-specific. Contact your system administrator for assistance.

Run Termination and Exit Status

The following features apply to automatically submitted runs that are controlled by the `STARRC_DP_STRING` command or the `STARRC_DP_STRING` environment variable:

- You can terminate runs by entering Ctrl+C on the command line.
- You can terminate a supervisor process that is running in the background by using the `kill` command.
- If you kill the supervisor process, or if it terminates abnormally, associated remote worker processes are automatically terminated.
- When all tasks in a StarRC run are complete, unneeded pending jobs are terminated.
- The supervisor process exits with an exit status of 0 for successful termination; nonzero values indicate unsuccessful termination.
- The standard output stream displays information such as the start and end times of each task and the final completion status.

Feedback

Support For Internet Protocol Versions IPv4 and IPv6

The StarRC tool can use either the IPv4 (32-bit) or IPv6 (128-bit) addressing protocol. The tool automatically detects the addressing mode on the submit host (the host used to launch jobs). The following usage notes apply:

- If the submit host supports both IPv4 and IPv6, you must set the `ENABLE_IPV6` command to `YES` to use IPv6 or `NO` to use IPv4. If you omit the command and the submit host supports both address modes, the StarRC tool issues an error message and stops.
- If the `STARRC_DP_STRING` command or environment variable specifies a list of machine names, the StarRC tool checks the mode of each host before submitting the jobs. If any host is incompatible with the submit host, the tool issues an error message and stops.
- If the job is submitted to a compute farm, all hosts must support the address mode of the submit host. Remote jobs that land on an incompatible host fail.
- If the submit host supports only IPv4 or only IPv6, do not use the `ENABLE_IPV6` command because the StarRC tool detects the address mode.

Methods For Specifying the Login Protocol

For all computing platforms, you must specify the login protocol with one of these methods:

- Set the `STARRC_DP_STRING` environment variable before launching the StarRC tool. Enclose the argument in single quotation marks because it might contain multiple items. For example:

  ```
  % setenv STARRC_DP_STRING 'list rsh alpha:2 beta:4 gamma'
  ```

- Specify the `STARRC_DP_STRING` command in the StarRC command file. For example:

  ```
  STARRC_DP_STRING: list rsh alpha:2 beta:4 gamma
  ```

If both the `STARRC_DP_STRING` command and the `STARRC_DP_STRING` environment variable are set, the StarRC command takes precedence.

Supported Computing Platforms

Distributed processing is available for the following computing environments:

- Single Host
- General Network of Hosts
- LSF System

Feedback

- Univa Grid Engine
- Runtime Design Automation System

Single Host

You can execute multiple StarRC runs on the host that you are logged into (the localhost). To run on a single host, the syntax is as follows:

```
list localhost:N
```

The argument `N` must be an integer less than or equal to the number of cores specified in the `NUM_CORES` command. For example:

```
NUM_CORES: 10
STARRC_DP_STRING: list localhost:8
```

General Network of Hosts

For a general network of hosts, the syntax is as follows:

```
list [login_protocol] host1[:n1] [host2[:n2] ... hostm[:nm]]
```

The arguments are as follows:

- The `login_protocol` argument is either `rsh` (the default) or `ssh`.

  Login access without a password to each host must be possible using the specified login protocol. Contact your system administrator to verify that the specified login protocol is permitted in your computing environment. For example, remote login using the `rsh` command might be prohibited. In this case, use the `ssh` protocol instead. Using a prohibited login protocol results in StarRC run termination.

- The argument `host1:n1` means that you are submitting n1 runs (an integer number of runs) to the machine with name host1. The default number of runs per machine is 1. Do not use spaces inside the *`host:n`* syntax, but use one or more spaces between hosts.

  The keyword for the host where the parent run starts is `localhost`. If you use `localhost`, use system calls instead of `rsh` to submit the runs.

This example for a general network uses the ssh protocol and submits 4 runs on system alpha, 2 runs on system beta, and 1 run on system gamma:

```
STARRC_DP_STRING: list ssh alpha:4 beta:2 gamma
```

Feedback

To configure `ssh` to work with the `STARRC_DP_STRING` command, use the following procedure. If you need assistance, contact your system administrator.

1. In your home directory, determine whether a .ssh directory already exists. If so, rename it or remove it.
2. Create a new directory named .ssh.
3. In the .ssh directory, run the following command to generate an authentication key:

   ```
   /usr/bin/ssh-keygen -f id_rsa -N "" -t rsa -q
   ```

   There is no space between the quotation marks.
4. In the .ssh directory, run the following command to copy the authentication key:

   ```
   cp id_rsa.pub authorized_keys
   ```
5. Modify the permissions for the home and .ssh directories as follows:

   ```
   chmod 755 $HOME
   chmod 700 $HOME/.ssh
   ```
6. Authorize a host for the first time as follows:

   ```
   ssh hostname
   ```

   System messages similar to the following appear, where (a.b.c.d) represents an IP address and 3f:5e... represents a full key fingerprint:

   ```
   The authenticity of host 'hostname (a.b.c.d)' can't be established.
   RSA key fingerprint is 3f:5e...
   Are you sure you want to continue connecting (yes/no)?
   ```
7. Answer `yes`. You should now be able to log in without a password or other intervention.
8. Repeat steps 6 and 7 for all hosts that you plan to use.

LSF System

The submission command for LSF systems is `bsub`. For example:

```
STARRC_DP_STRING: bsub -R "rusage[mem=5000]"
```

Univa Grid Engine

The submission command for Univa Grid Engine (UGE) systems (formerly known as Oracle Grid Engine) is `qsub`. For example:

```
STARRC_DP_STRING: qsub -P bnormal -l "mem_free=1G" \
  -v "STARRC_LICENSE_WAIT=YES"
```

To pass environment variables to the job, you must use the `-v` option. In this example, the `STARRC_LICENSE_WAIT` environment variable is set to allow license queuing.

Feedback

Runtime Design Automation System

The submission command for a Runtime Design Automation (RTDA) system is `nc run`. For example:

```
% setenv STARRC_DP_STRING 'nc run -R "rusage[mem=5000]"'
```

Distributed Processing Error and Message Handling

A StarRC job might encounter an error and stop. In this case, the following actions take place:

- The tool issues an error message to the standard error stream and to the summary file.
- All processes running concurrently at the time of failure run to completion. This practice helps you to find additional issues in the design or extraction setup.
- The tool does not start any new tasks after an error is encountered in any concurrently running process.

For StarRC runs that use the `STARRC_DP_STRING` command, information and warning messages generated by interprocess communication utilities are saved in files located in the star/dp_logs/star_logs directory. Information in these files is intended only for diagnosing problems with distributed processing.

In some computing environments, Synopsys Common Licensing (SCL) license checkout information is also written to files in the star/dp_logs/st-ar_logs directory.

Distributed Processing Reports

After the run is complete, the StarRC tool provides a report in the star/summary file. The report includes the following information:

- Stage summary

  Reports runtime, memory consumption, CPU or host on which the job was executed, and job completion timestamp.

- Distributed processing summary for distributed stages

  Shows the maximum and average runtime for each CPU.

- Total runtime

  Shows timestamps for the beginning and the end of the run, in addition to the total runtime.

- Pre-extraction and postextraction times

Reports the runtime information of pre-extraction and postextraction stages.

- Information, warning, and error messages

The following example shows the distributed process summary report based on cores:

```
CPU_03 |*xTractDB  Elp=01:22:28 Cpu=01:22:25 Mem=2394.9 | Feb 28 14:50:17
 | sys701

CPU_01 |*xTractDB  Elp=01:23:13 Cpu=01:23:10 Mem=2332.2 | Feb 28 14:51:00
 | sys701

CPU_02 |*xTractDB  Elp=01:23:22 Cpu=01:23:19 Mem=2180.9 | Feb 28 14:51:14
 | sys702

CPU_04 |*xTractDB  Elp=01:23:42 Cpu=01:23:35 Mem=2431.6 | Feb 28 14:51:23
 | sys702
```

Status of Distributed Processing

The StarRC tool provides the report in the following files:

- star/summary file: Located in the working directory that includes all errors for distributed processing runs.
- master.log file: Located in the star directory that includes progress of all worker processes reporting to the supervisor process. The subdirectories in the star directory include many worker and error log files.

To read the files generated during a StarRC run in distributed processing mode, run the distributed processing tracer (DP-Tracer.py) script any time on these files to

- Read through all log files to know the current status and read through all summary files for errors
- Display information about remote processors, execution status of all running tasks, and locations of files for all errors reported by the tool

Running the Distributed Processing Tracer Script

To run the distributed processing tracer (DP-Tracer.py) script, use the following command:

```
$./DP-Tracer [-flags] <star_cmd>
```

However, if the Python path in your environment is different and the tool issues the `DP-Tracer.py: Command not found` error message, use the following command:

```
$python DP-Tracer <star_cmd>
```

Feedback

Table 10 shows how to run the script to locate the path of the StarRC command file to find the required files. You can run the script from the same directory as the StarRC run in distributed processing mode.

The script reads through the following files to locate and provide information on tasks and errors:

```
<block/star_directory>.star_sum
<star_directory>/ <block/star_directory>.star_sum_cpu_#
<star_directory>/dp_logs/star_logs/master.log
<star_directory>/dp_logs/star_logs/worker_#.log
<star_directory>/summary/*
```

The tool prints output of the script to a console and writes to the *dp_tracer.log file* in the current directory.

Table 10 Extracting the Path of StarRC Command File by the DP-Tracer.py Script

At the default path	Using the -StarXtract_location flag
If the `StarXtract` command is run in the same directory as the StarRC command File, the script extracts the path of the StarRC command File to find the required files.	If the `StarXtract` command is run in a different directory than the StarRC command File, the script uses the path specified with the `-StarXtract_location` flag to extract the StarRC command File and finds the required files.
For example, if you run the `DP-Tracer test1/star_cmd` command from the star directory (the directory specified by the `STAR_DIRECTORY: star` command), the tool finds the test1/star file.	For example, if you run the `DP-Tracer test1/star_cmd -StarXtract_location test2/run_dir` command from the star directory (the directory specified by the `STAR_DIRECTORY: star` command), the tool finds the test2/run_dir/star file.

The examples show the following reports that are generated when you use the distributed processing tracer (DP-Tracer.py) script:

- Example 1: Ongoing Run Without Errors
- Example 2: Completing a Successful Run
- Example 3: Completing an Unsuccessful Run
- Example 4: Progress of Starved Worker Process
- Example 5: Losing Connection by a Worker Process Unexpectedly
- Example 6: Using the -show_all_worker_tasks Flag
- Example 7: Using the -filter Flag

Feedback

Example 1 Ongoing Run Without Errors

```
time stamp: 1570148553.7
-------------------------------------------------------------------------
|host_name| used cpu| idle cpu| ram            | disk
-------------------------------------------------------------------------
|abc2     | 33.3%   | 66.7%   | 54/252 GB used | 1030/1380(74.7%) GB used
|abc5     | 0.0%    | 100.0%  | 105/252 GB used| 1030/1380(74.7%) GB used
|abc3     | 0.0%    | 100.0%  | 28/252 GB used | 1030/1380(74.7%) GB used
|abc8     | 0.0%    | 100.0%  | 30/252 GB used | 1030/1380(74.7%) GB used
-------------------------------------------------------------------------

Currently Running
-------------------------------------------------------------------------
|Worker ID|CPU_ID |host|processID|jobID   |Task               | Time   |
                  _name
-------------------------------------------------------------------------
|worker 3 |cpu 1  |abc3|30760    |7100515 |Translate_part1 DB | 0.77sec|
|worker 4 |cpu 4  |abc8|5257     |7100506 |Translate_part2 DB | 0.05sec|
|worker 1 |cpu 3  |abc2|53103    |7100514 |Translate_part3 DB | 0.00sec|
|worker 2 |cpu 2  |abc5|20298    |7100507 |None               |        |
-------------------------------------------------------------------------

No Reported Errors.
```

Example 2 Completing a Successful Run

```
time stamp: 1570145726.62
-------------------------------------------------------------------------
|host|used |idle | ram               | disk                          |
_name  cpu   cpu
-------------------------------------------------------------------------
|abc3|6.7% |93.2%| 6354/386203 MB used| 1411990/1419306 (99.5%) MB used |
|abc2|17.0%|83.0%| 5736/192295 MB used| 1411990/1419306 (99.5%) MB used |
-------------------------------------------------------------------------

Run Successful
No Tasks Currently Running
-----------------------------------------------------------------
|Worker ID |CPU_ID |host Name |processID |jobID    |Task    |Time |
-----------------------------------------------------------------
|worker 1  |cpu 2  |abc2      |206044    |2690255  |None    |     |
|worker 2  |cpu 4  |abc3      |60341     |2690267  |None    |     |
|worker 3  |cpu 3  |abc3      |60338     |2690256  |None    |     |
|worker 4  |cpu 1  |abc2      |206039    |2690254  |None    |     |
-----------------------------------------------------------------

No Reported Errors.
```

Example 3 Completing an Unsuccessful Run

```
time stamp: 1570146571.53
-------------------------------------------------------------------------
|host|used cpu|idle cpu|ram               | disk                       |
```

```
_name
-------------------------------------------------------------------------
| abc1|38.9%   |61.1%   |397/18948 MB used | 1030/1380 (74.7%) GB used |
| abc9|54.8%   |45.2%   |47/188 GB used    | 1030/1380 (74.7%) GB used |
| abc3|30.0%   |70.0%   |55/505 GB used    | 1030/1380 (74.7%) GB used |
| abc2|34.8%   |65.2%   |15/252 GB used    | 1030/1380 (74.7%) GB used |
-------------------------------------------------------------------------

Run not Successful
No Tasks Currently Running
-----------------------------------------------------------------
|Worker ID |CPU_ID  |host Name |processID |jobID   |Task |Time  |
-----------------------------------------------------------------
|worker 1  |cpu 1   |abc11     |30760     |7060681 |None |      |
|worker 2  |cpu 2   |abc79     |5257      |7060682 |None |      |
|worker 3  |cpu 3   |abc10     |53103     |7060687 |None |      |
|worker 4  |cpu 4   |abc20     |20298     |7060683 |None |      |
-----------------------------------------------------------------

Reported Errors
-------------------------------------------------------------------------
|Task       | Error                  | Error Origin    | Logs           |
-------------------------------------------------------------------------
|XinCombine | ERROR: Internal system | CERT_IO.star_sum| star/summary/Xi|
|           | error, cannot recover. |                 | nCombine.sum   |
|           | ERROR: StarRC task     |                 |                |
|           | XinCombine failed. For |                 |                |
|           | more details, see task |                 |                |
|           | summary file           |                 |                |
|           | 'XinCombine.sum'.      |                 |                |
|           |  (SX-0255)             |                 |                |
-------------------------------------------------------------------------
```

Example 4 Progress of Starved Worker Process

```
time stamp: 1570148673.75
**********************************
Warning: 1 out of 4 workers pending due to unavailable cores
        This may lead to later than expected compleation time.
        Job names:
           starrc11-p001.j07f5f52a90408b42.0003
**********************************
-------------------------------------------------------------------------
| host| used cpu | idle cpu | ram             | disk
 _name
-------------------------------------------------------------------------
| abc2 | 21.3%    | 78.7%    | 55/252 GB used  | 1030/1380(74.7%) GB used
| abc5 | 20.0%    | 80.0%    | 105/252 GB used | 1030/1380(74.7%) GB used
| abc8 | 16.3%    | 83.7%    | 31/252 GB used  | 1030/1380(74.7%) GB used
-------------------------------------------------------------------------

Run Successful
No Tasks Currently Running
```

Feedback

```
-------------------------------------------------------------------------
|Worker ID |CPU_ID    |host Name|processID |jobID     |Task      |Time |
-------------------------------------------------------------------------
|worker 1  |cpu 3     |abc2     |5256      |7100513   |None      |     |
|worker 2  |cpu 2     |abc5     |20288     |7100508   |None      |     |
|worker 4  |cpu 4     |abc8     |53103     |7100518   |None      |     |
-------------------------------------------------------------------------

No Reported Errors.
```

Example 5 Losing Connection by a Worker Process Unexpectedly

```
time stamp: 157022377.38
-------------------------------------------------------------------------
| host| used cpu|idle cpu|ram                | disk
-------------------------------------------------------------------------
| abc1| 38.9%   |61.1%   |3972/18948 MB used | 1030/1380 (74.7%) GB used
| abc9| 54.8%   |45.2%   |47/188 GB used     | 1030/1380 (74.7%) GB used
| abc3| 30.0%   |70.0%   |55/505 GB used     | 1030/1380 (74.7%) GB used
| abc2| 34.8%   |65.2%   |15/252 GB used     | 1030/1380 (74.7%) GB used
-------------------------------------------------------------------------

Run not Successful
-------------------------------------------------------------------------
|Worker ID|CPU_ID |host Name  |processID |jobID   |Task |Time         |
-------------------------------------------------------------------------
|worker 1 |cpu 1  |abc1       |18038     |7060588 |None |disconnected |
|worker 2 |cpu 2  |abc9       |282734    |7060580 |None |             |
|worker 3 |cpu 3  |abc3       |28264     |7060581 |None |             |
|worker 4 |cpu 4  |abc2       |89397     |7060586 |None |             |
-------------------------------------------------------------------------

Reported Errors
-------------------------------------------------------------------------
|Task       |Error                      |Error Origin    |Logs
-------------------------------------------------------------------------
|N/A        |ERROR: worker is no longer|worker_001.log  |
|           |connected to master,       |                |
|           |killing child process.     |                |
|           | Host abc1                 |                |
|           |                           |                |
|XinCombine |ERROR: Internal system     |CERT_IO.star_sum|star/summary/Xi
|           |error, cannot recover.     |                |nCombine.sum
|           |ERROR: StarRC task         |                |
|           |XinCombine failed. For     |                |
|           |more details, see task     |                |
|           |summary file               |                |
|           |'XinCombine.sum'.(SX-0255)|                |
-------------------------------------------------------------------------
```

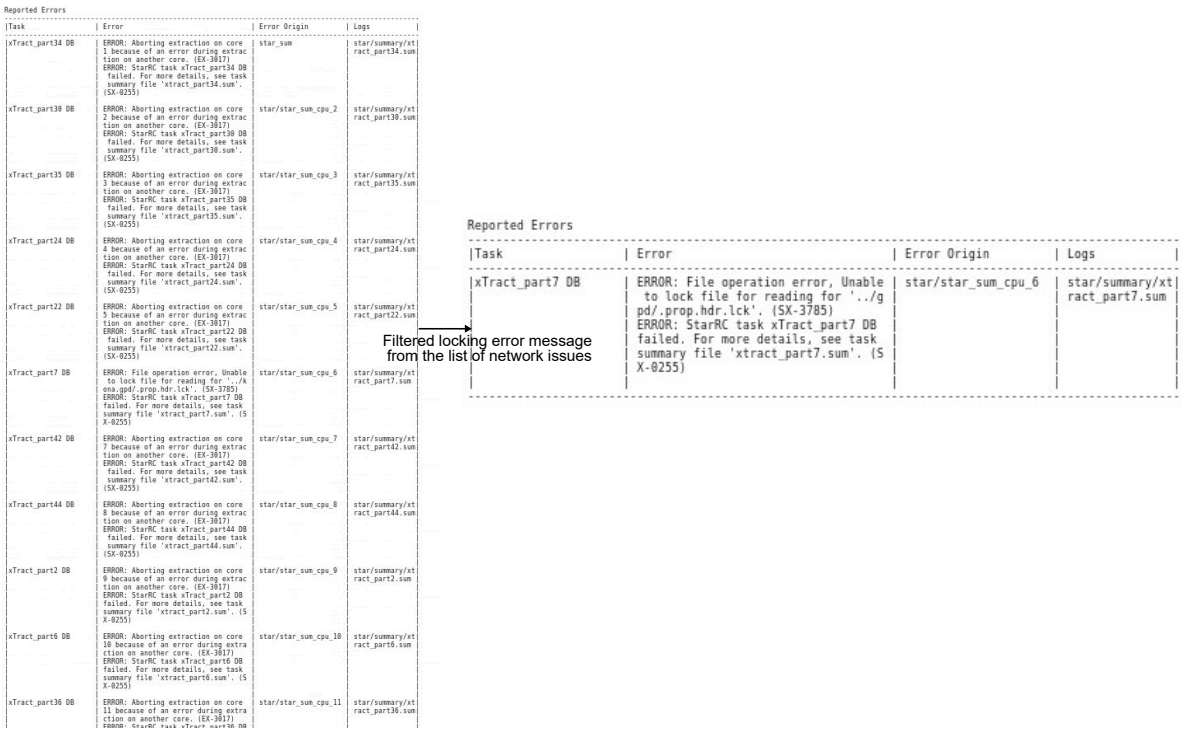
Feedback

Example 6 Using the -show_all_worker_tasks Flag

```
-------------------------------------------------------------------
|worker 1                      |Elp       |Cpu       |Mem       |
-------------------------------------------------------------------
|Split Top Def_part1           |00:00:04  | 00:00:00 |357.3     |
|Process Standard Cells_part1  |00:00:00  | 00:00:00 |359.3     |
|Translate_part3 DB            |00:00:00  | 00:00:00 |355.2     |
|Merge GPD Instance Pins       |00:00:01  | 00:00:00 |356.9     |
|Netlist Assembly Setup        |00:00:00  | 00:00:00 |387.1     |
|xTract_part1 DB               |00:00:00  | 00:00:00 |357.4     |
|Stitch_part1 DB               |00:00:07  | 00:00:04 |527.3     |
|Report Violations             |00:00:00  | 00:00:00 |247.0     |
|Netlist Assembly4 DB          |00:00:00  | 00:00:00 |352.3     |
|GPD Converter Part 2          |00:00:00  | 00:00:00 |247.4     |
|GPD Converter Part 7          |00:00:01  | 00:00:00 |251.6     |
-------------------------------------------------------------------

-----------------------------------------------------------
|worker 2                |Elp       |Cpu       |Mem       |
-----------------------------------------------------------
|Split Top Def_part2     |00:00:04  | 00:00:00 |357.3     |
|Process Top Cell_part2  |00:00:00  | 00:00:00 |359.3     |
|Translate_part2 DB      |00:00:01  | 00:00:00 |398.7     |
|XinCombine              |00:00:01  | 00:00:00 |398.0     |
|GPD XtractSetup         |00:00:00  | 00:00:00 |352.2     |
|Clean Directory         |00:00:00  | 00:00:00 |397.5     |
|xTract_part3 DB         |00:00:00  | 00:00:00 |352.3     |
|Stitch Pre Process 2    |00:00:05  | 00:00:03 |514.1     |
|xTract Post Process DB  |00:00:00  | 00:00:00 |244.9     |
|Netlist Assembly3 DB    |00:00:00  | 00:00:00 |247.8     |
|GPD Converter Part 1    |00:00:00  | 00:00:00 |246.8     |
|GPD Converter Part 6    |00:00:01  | 00:00:00 |251.7     |
|GPD Converter Merge     |00:00:00  | 00:00:00 |245.5     |
-----------------------------------------------------------
```

Example 7 Using the -filter Flag

Feedback



Reported Errors

Task	Error	Error Origin	Logs
xTract_part34 DB	ERROR: Aborting extraction on core 1 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part34 DB failed. For more details, see task summary file 'xtract_part34.sum'. (SX-0255)	star_sum	star/summary/xtract_part34.sum
xTract_part30 DB	ERROR: Aborting extraction on core 2 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part30 DB failed. For more details, see task summary file 'xtract_part30.sum'. (SX-0255)	star/star_sum_cpu_2	star/summary/xtract_part30.sum
xTract_part35 DB	ERROR: Aborting extraction on core 3 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part35 DB failed. For more details, see task summary file 'xtract_part35.sum'. (SX-0255)	star/star_sum_cpu_3	star/summary/xtract_part35.sum
xTract_part24 DB	ERROR: Aborting extraction on core 4 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part24 DB failed. For more details, see task summary file 'xtract_part24.sum'. (SX-0255)	star/star_sum_cpu_4	star/summary/xtract_part24.sum
xTract_part22 DB	ERROR: Aborting extraction on core 5 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part22 DB failed. For more details, see task summary file 'xtract_part22.sum'. (SX-0255)	star/star_sum_cpu_5	star/summary/xtract_part22.sum
xTract_part7 DB	ERROR: File operation error, Unable to lock file for reading for '../kona.gpd/.prop.hdr.lck'. (SX-3785) ERROR: StarRC task xTract_part7 DB failed. For more details, see task summary file 'xtract_part7.sum'. (SX-0255)	star/star_sum_cpu_6	star/summary/xtract_part7.sum
xTract_part42 DB	ERROR: Aborting extraction on core 7 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part42 DB failed. For more details, see task summary file 'xtract_part42.sum'. (SX-0255)	star/star_sum_cpu_7	star/summary/xtract_part42.sum
xTract_part44 DB	ERROR: Aborting extraction on core 8 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part44 DB failed. For more details, see task summary file 'xtract_part44.sum'. (SX-0255)	star/star_sum_cpu_8	star/summary/xtract_part44.sum
xTract_part2 DB	ERROR: Aborting extraction on core 9 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part2 DB failed. For more details, see task summary file 'xtract_part2.sum'. (SX-0255)	star/star_sum_cpu_9	star/summary/xtract_part2.sum
xTract_part6 DB	ERROR: Aborting extraction on core 10 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part6 DB failed. For more details, see task summary file 'xtract_part6.sum'. (SX-0255)	star/star_sum_cpu_10	star/summary/xtract_part6.sum
xTract_part36 DB	ERROR: Aborting extraction on core 11 because of an error during extraction on another core. (EX-3017) ERROR: StarRC task xTract_part36 DB	star/star_sum_cpu_11	star/summary/xtract_part36.sum

Filtered locking error message from the list of network issues

Reported Errors

Task	Error	Error Origin	Logs
xTract_part7 DB	ERROR: File operation error, Unable to lock file for reading for '../gpd/.prop.hdr.lck'. (SX-3785) ERROR: StarRC task xTract_part7 DB failed. For more details, see task summary file 'xtract_part7.sum'. (SX-0255)	star/star_sum_cpu_6	star/summary/xtract_part7.sum

3

Design Databases

This chapter describes how to manage the design and layout-versus-schematic databases that the StarRC tool uses as input.

For more information, see the following topics:

- Introduction to Physical Databases
- The Fusion Compiler or IC Compiler II Flow
- The IC Compiler (Milkyway) Database Flow
- The LEF/DEF Database Flow
- The Calibre Connectivity Interface Flow
- The Hercules LVS Tool Flow
- The IC Validator LVS Tool Flow
- Cross-Referencing in Transistor-Level Flows
- Parameterized Cells
- Metal Fill

Feedback

Introduction to Physical Databases

The StarRC tool calculates parasitic effects based on the physical layout of a design. The output of a layout-versus-schematic (LVS) tool is used to determine where the physical parasitic devices connect to the logical elements in the design. The supported databases are as follows:

- Design libraries in the New Data Model (NDM) format created by the Fusion Compiler or IC Compiler II tool
- Design libraries in the Milkyway format created by the IC Compiler tool
- Design libraries in LEF/DEF format
- LVS output from the IC Validator tool
- LVS output from the Hercules tool
- LVS output from the Mentor Graphics Calibre tool

The StarRC tool can run on layouts containing opens or shorts. Special provisions are made to repair the netlist so that delay calculation can be performed even on the problem nets. Shorted nets are always listed separately in the output netlist and contain only the parasitics for the polygons that belong to them. You can optionally connect open nets by using the `NETLIST_CONNECT_OPENS` command. Both opens and shorts are reported explicitly in detailed summary files. However, some StarRC results might be inaccurate in a design with shorts or opens.

The StarRC tool integrates directly with the Synopsys IC Compiler, IC Compiler II, or Fusion Compiler tool for signoff design closure. The tool also integrates smoothly with LEF/DEF-based place and route flows. You can read leaf cells in GDSII format directly to merge them during translation into the layout database.

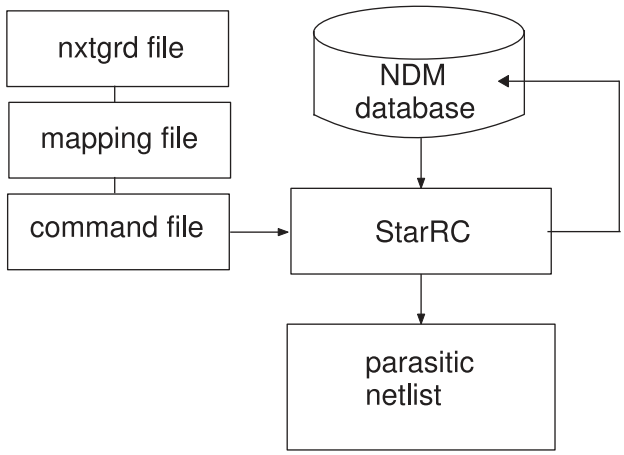
Feedback

The Fusion Compiler or IC Compiler II Flow

The Fusion Compiler or IC Compiler II tool generates design libraries in the NDM format. The StarRC tool reads these design libraries directly.

The extraction flow shown in Figure 9 shows the Fusion Compiler or IC Compiler II layout database flow. The layout does not need to be LVS-clean for completion of a successful extraction. The StarRC tool issues warnings when it finds open or shorted nets.

Figure 8 NDM Format Design Library Flow



Examples of commands that are used only for Fusion Compiler or IC Compiler II design libraries are as follows:

- `NDM_CELL_REPORT_FILE`
- `NDM_DATABASE`
- `NDM_DESIGN_VIEW`
- `NDM_LAYOUT_VIEW`
- `NDM_EXPAND_HIERARCHICAL_CELLS`

Via Ladder Support

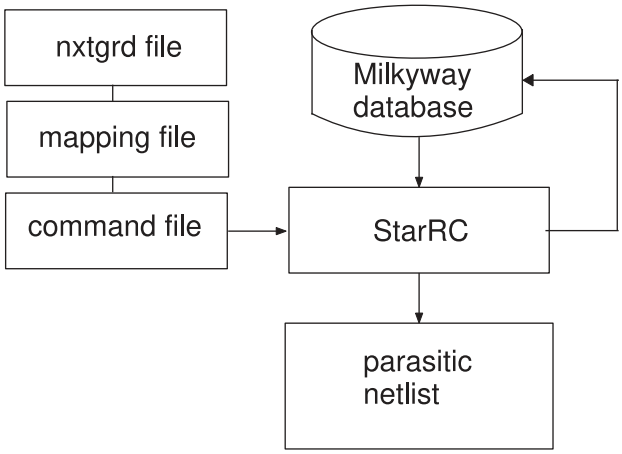
The StarRC tool supports via ladders (also known as via pillars) in designs created by the Fusion Compiler or IC Compiler II tool in both NDM and LEF/DEF formats. In LEF/DEF designs, via ladders are marked with the prefix NR_STACK. Multiple mask patterning is supported for via pillars.

Feedback

The IC Compiler (Milkyway) Database Flow

The extraction flow shown in Figure 9 shows the Milkyway layout database containing all network annotation information. Milkyway databases can be read directly by the StarRC tool. The layout does not need to be LVS-clean for completion of a successful extraction. The StarRC tool issues warnings when it finds open or shorted nets.

Figure 9 Milkyway Design Extraction Flow



Examples of commands that are used only for Milkyway design libraries are as follows:

- MILKYWAY_ADDITIONAL_VIEWS
- MILKYWAY_CELL_VIEW
- MILKYWAY_DATABASE
- MILKYWAY_EXPAND_HIERARCHICAL_CELLS
- MILKYWAY_EXTRACT_VIEW
- MILKYWAY_REF_LIB_MODE
- MILKYWAY_SHOW_CELL_INFO_DETAIL

When determining the reference library status of a Milkyway database, the StarRC tool uses the reference library mode stored within the main Milkyway database with the Milkyway `dbSetLibRefCtrlFileMode` function. This function specifies whether the reference library tree source is from the Internal Reference Mode or the Reference Control File Mode. See the Milkyway documentation for more information.

Feedback

Antenna Diodes in Milkyway Designs

In the Milkyway database, when an antenna diode cell is inserted by place and route tools, the diode cell type is automatically set to a standard filler type. The StarRC tool detects standard filler cell types automatically and ignores them in the output parasitic file.

However, some antenna diode cells might be inserted manually with an incorrect cell type. This causes the StarRC tool to extract and netlist the incorrectly-defined diode cells. Diode cells should not be listed in the output parasitics file because they are not part of the original Verilog or SPICE netlist. They create parasitic back-annotation errors and warnings in the PrimeTime tool.

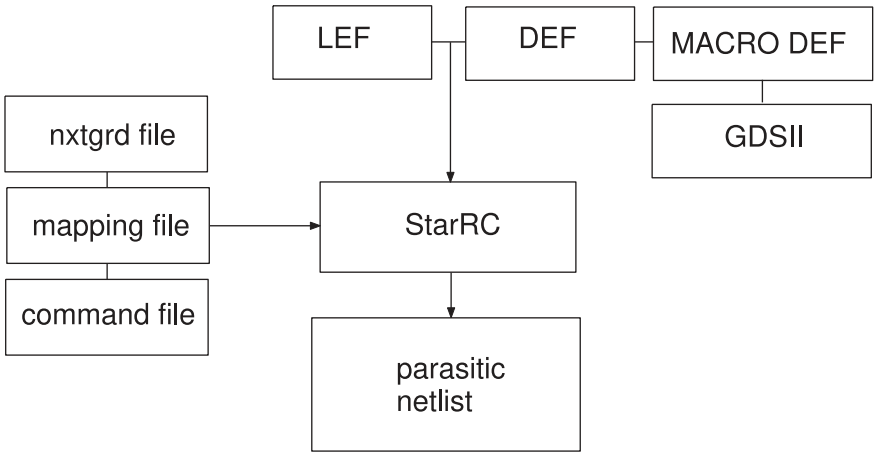
To correct a cell definition, query the diode cell instances to confirm the CELL FLAG property, set the antenna cell type to *standard filler*, and convert the pin type to DIODE-PIN type.

Feedback

The LEF/DEF Database Flow

Figure 10 shows the LEF/DEF design flow. The Library Exchange Format/Design Exchange Format (LEF/DEF) layout description is read directly by the StarRC tool. LEF/DEF file format versions 5.2 through 5.8 are supported.

Figure 10 LEF/DEF Extraction Flow



Most information about the design is read directly from the LEF/DEF database. The StarRC tool automatically identifies the following:

- Power nets
- Primary input and output ports
- The top-level block
- Skip cells

Examples of commands that are used only for LEF/DEF design libraries are as follows:

- `DEF_ATTRIBUTE_FROM_LEF`
- `DEF_USE_PINS`
- `LEF_FILE`
- `LEF_USE_OBS`
- `MACRO_DEF_FILE`
- `TOP_DEF_FILE`

Feedback

Hierarchical LEF/DEF designs are fully supported. You can specify multiple MACRO_DEF_FILE commands along with a single TOP_DEF_FILE command to extract a hierarchically routed design. If a macro DEF file is specified, all subcells referenced in the macro DEF must have corresponding MACRO definitions within the library LEF files.

You do not need to provide LEF descriptions for DEF macros. For DEF macros, the macro details are taken from the DEF file, not the LEF file. The DEF file is assumed to be more accurate than the LEF file.

The DEF_USE_PINS command controls the selection of pin information. If the command is set to YES (the default), pin information is taken preferentially from the DEF file. Pins not found in the DEF file are taken from the LEF file. Conversely, if the DEF_USE_PINS command is set to NO, the pin information is taken only from the LEF file.

The order in which the LEF files are specified is important. The technology LEF that contains layer definitions is required before any of the library LEF files are read. You can choose to obtain the routing width information for a specific DEF macro from any LEF file by using the DEF_ATTRIBUTE_FROM_LEF command.

Parasitic capacitance information contained in the LEF files is ignored. The layer resistance specifications are used only if resistance information is missing from both the nxtgrd file and the mapping file.

The LEF file must always contain via definitions, even if the vias are redefined in the DEF file. The DEF description takes precedence in the extraction whereas the LEF description is used for layer mapping and initial connectivity.

Merging Library GDSII Files in LEF/DEF Designs

GDSII information can be directly merged into the LEF description for library cells or macro blocks. When you use this feature, the StarRC tool uses only the pin shapes from the LEF MACRO cell definitions and discards the obstructions and other objects. The actual physical layout from the GDSII library is translated and used to represent the contents of skip cells during extraction.

GDSII layers for inclusion must be equated to a LEF database layer with the GDS_LAYER_MAP_FILE command. If any GDSII layer is not specified in the map file, it is not translated for extraction and does not contribute to the parasitics.

Indexes Warning in the Netlist Stage

If there is no port geometry for the pins of the cells in a LEF file, a warning is issued. You must make the necessary correction in the LEF file. For example:

```
MACRO  inv
  PIN a
    DIRECTION INPUT ;
  END a
```

Feedback

```
  PIN y
    DIRECTION OUTPUT ;
  END y
END  inv
```

Translation of Routing DEF Blockages

In a design flow, you can divide the chip into blocks and perform the routing of the blocks separately. These blocks are then integrated at the top level. While carrying out the routing at the block level, you can define routing blockages that represent the tracks in which the top-level routing resides.

In addition to performing worst-corner (most pessimistic) extraction at the block or macro level, you can also consider the effect of the top-level signal net on the parasitics of the block-level nets. The top-level routing information might not be available during the block-level extraction, therefore the blockages defined for the macro serve as an approximation for the top-level routing.

By default, the StarRC tool does not read or translate DEF blockages. Set the `TRANSLATE_DEF_BLOCKAGE` command to `YES` to translate the routing DEF blockages from the file specified by the `TOP_DEF_FILE` command. DEF blockages from files specified by the `MACRO_DEF_FILE` command are ignored. The routing information corresponding to these blockages is already present in the top DEF file. Placement blockages in the top DEF file are ignored.

The StarRC tool does not translate zero spacing OBS features in LEF files.

The following is an example of the blockage section in a DEF file:

```
[BLOCKAGES numBlockages ;
    [- LAYER layerName
        [+ COMPONENT compName | + SLOTS | + FILLS | + PUSHDOWN]
        [+ SPACING minSpacing | + DESIGNRULEWIDTH effectiveWidth]
        {RECT pt pt | POLYGON pt pt pt …} …
    ;] …

[- PLACEMENT
    [+ COMPONENT compName | + PUSHDOWN]
    {RECT pt pt} …
    ;] …
```

Reading Line-End Extension Blockages

A line-end extension blockage is a marker shape. The StarRC tool modifies the overlap area between a design shape and a marker shape before extraction. The StarRC tool reads or translates line-end extension blockages only if they are defined in both the LEF and DEF files and a line-end extension table is defined on the corresponding ITF layer in an ITF file, even if the `TRANSLATE_DEF_BLOCKAGE` command is set to `NO`.

Feedback

To read and translate the routing line-end extension blockages in the LEF and DEF files, specify the files with the TOP_DEF_FILE or MACRO_DEF_FILE command. Any blockages on a blockage layer are recognized as a line-end extension blockage on the corresponding routing layer. By default, the tool does not read or translate the line-end extension blockages.

Example 8 and Example 9 show the line-end extension blockage section in the LEF and DEF files:

Example 8 Line-End Extension Blockage Layer Defined to Map With a Routing Layer in a LEF File

```
…
LAYER M1
  TYPE ROUTING ;
  MASK 4 ;
  WIDTH 0.1 ;
END M1

LAYER LEE_BLK_1
  TYPE MASTERSLICE ;
  PROPERTY LEF58_TYPE " TYPE RCBLOCKAGE M1 ; " ;
END LEE_BLK_1
…
```

Example 9 Line-End Extension Blockage Layers Defined in a DEF file

```
…
BLOCKAGES 5 ;
  LAYER LEE_BLK_1
    RECT ( 0  0 ) ( 100 100 ) ;

  LAYER M1 + MASK 4
    RECT ( 500 500 ) ( 400 400 ) ;

  LAYER LEE_BLK_1 + MASK 2
    RECT ( 200 300 ) ( 300 200 )
    RECT ( 1000 -1000 ) ( -1000 1000 ) ;

  LAYER M1 + MASK 1
    RECT ( 100 100 ) ( 200 200 ) ;

LAYER LEE_BLK_1 + MASK 3
    RECT ( 300 300 ) ( 400 400 ) ;

END BLOCKAGES
…
```

Feedback

Note:

The tool issues warning messages if a layer of blockage is defined as `TYPE MASTERSLICE` without a `PROPERTY` statement for the line-end extension blockage.

Make sure that the LEF blockages are defined on the corresponding ITF layer in the ITF file for the tool to read or translate the line-end extension blockages. The `LINE_END_EXTENSION_TABLE` syntax in the ITF file is as follows:

```
LINE_END_EXTENSION_TABLE options {
   PERPENDICULAR_TO_REFERENCE I PARALLEL_TO_REFERENICE {
   NUMBER_OF_WIDTH1S = num_of_width1s
      WIDTH1 =. <um> {
      SPACINGS { s1 s2 ... sm }
      WIDTH2   {w1 w2 ... wn}
      VALUES   { v1_s1_w1 v_s2_w1 ... v1_sm_w1
                 v1_s1_w2 v1_s2_w2 ... v1_sm_w2
                 ...
                 v1_s1_wn v1_s2_wn ... v1_sm_wn}
                 ...
               }
```

See Also

- LINE_END_EXTENSION_TABLE

Feedback

The Calibre Connectivity Interface Flow

The StarRC tool can read output files from the Calibre layout versus schematic (LVS) tool. This is achieved by using the Calibre Connectivity Interface, which generates layout, connectivity, port, and cross-referencing information from the Calibre LVS run in a form that is usable by the StarRC tool.

Using the Calibre Connectivity Interface flow, you can generate Detailed Standard Parasitic Format (DSPF) and Standard Parasitic Exchange Format (SPEF) netlists.

The Calibre tool generates multiple files with information about polygons, connectivity, text net names, device tables, and LVS cross-referencing tables. The StarRC tool reads the following Calibre Connectivity Interface information:

- LVS extraction report file

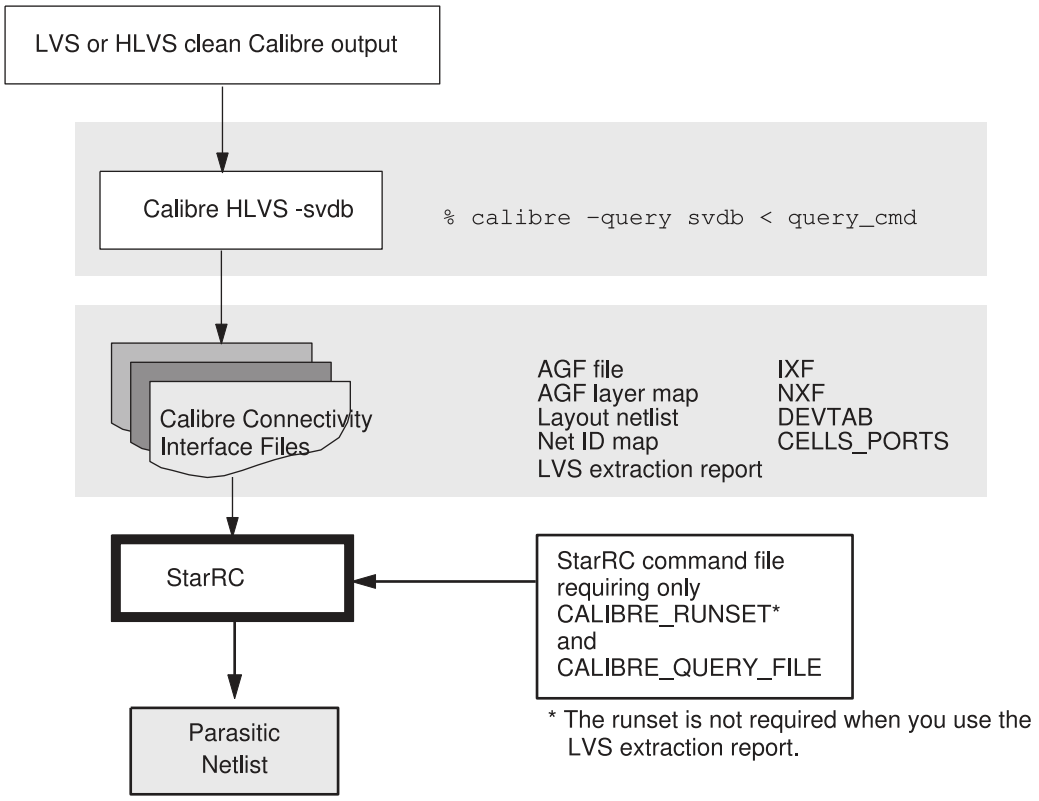
  When you include the LVS extraction report in the Calibre Connectivity Interface, the StarRC tool does not need to parse the Calibre rulefile.
- Annotated GDSII (AGF) file containing polygon and connectivity information
- Mapping file describing layer mapping between the AGF file and the Calibre runset
- Ideal layout netlist
- Layout net names file
- Calibre device table describing terminal layers for all possible devices in the runset
- Top-level ports file
- LVS net cross-referencing table
- LVS instance cross-referencing table
- Calibre runset to interpret layer connectivity

By reading the Calibre query command file directly, the StarRC tool can locate all Calibre Connectivity subordinate files relating to the ideal layout netlist, the annotated GDSII file, and the cross-reference information. Error checking on the Calibre query command file ensures that the Calibre Query Server was invoked with the required options for use with the StarRC tool.

Figure 11 illustrates the basic Calibre Connectivity Interface flow. The procedure varies depending on whether or not you use an LVS extraction report.

Feedback

Figure 11 Calibre Connectivity Interface Flow



Procedure Without an LVS Extraction Report

Follow these steps if your Calibre Connectivity Interface flow does not include an LVS extraction report file:

1. Generate an LVS or HLVS clean design.
2. Create a collection of Calibre Connectivity Interface input files with the query command file, named query_cmd in the following example:

   ```
   % calibre -query svdb < query_cmd
   ```

   The `svdb` file contains the design data from Calibre. The file named query_cmd contains the Calibre options that create the Calibre Connectivity Interface files.
3. Create a StarRC command file containing the `CALIBRE_RUNSET` and `CALIBRE_QUERY_FILE` commands. For example,

   ```
   BLOCK:  my_example
   TCAD_GRD_FILE:  my_example.nxtgrd
   ```

```
MAPPING_FILE:  my_example.map
CALIBRE_RUNSET:  calibre.runset
CALIBRE_QUERY_FILE: query_input
```

4. Run the prepared command file.

   The CALIBRE_RUNSET and CALIBRE_QUERY_FILE commands find the necessary Calibre file information and create the output file.

Procedure Using an LVS Extraction Report

If the Calibre query file uses the LVS SETTINGS REPORT WRITE command to write an extraction report, the StarRC CALIBRE_QUERY_FILE command automatically obtains the runset file and Push Down Back Annotation (PDBA) information from the query file. The CALIBRE_RUNSET and CALIBRE_PDBA_FILE commands are invalid in this case.

Note:

It is the responsibility of the Calibre tool to ensure that the information in the LVS extraction report is accurate and correctly handles directives and conditional statements.

Follow these steps if your Calibre Connectivity Interface flow includes an LVS extraction report file:

1. Generate an LVS or HLVS clean design.
2. Create a collection of Calibre Connectivity Interface input files with the query command file, named query_cmd in the following example:

   ```
   % calibre -query svdb < query_cmd
   ```

   The svdb file contains the design data from Calibre. The file named query_cmd contains the Calibre options that create the Calibre Connectivity Interface files.
3. Write a StarRC command file containing the CALIBRE_QUERY_FILE command but no CALIBRE_RUNSET or CALIBRE_PDBA_FILE commands.
4. (Optional) If you want to use an LVS extraction report file that is different from the one specified in the Calibre LVS SETTINGS REPORT WRITE command, specify the file name with the LVS_EXTRACTION_REPORT_FILE command in the StarRC command file.
5. Run the prepared command file.

Feedback

Error Conditions in StarRC-Calibre Flows

Selected StarRC error conditions related to the Calibre Connectivity Interface are as follows:

- The `GDS NETPROP NUMBER`, `GDS PLACEPROP NUMBER`, or `GDS DEVPROP NUMBER` properties must be set in accordance with the Calibre Query File requirements.
- `GDS SEED LAYER ORIGINAL`, when specified, must appear before the `GDS MAP` command in the Calibre query command file.
- `XREF:COMPLETE` is not supported in the Calibre Connectivity Interface flow.
- Duplicate layer mappings in the Calibre AGF layer mapping file produce an error condition because data in the AGF might be corrupted as a result. For example, if two AGF layers are mapped to the same layer number, the following error message appears during the Translate DB stage:

  ```
  ERROR:StarXtract
  ERROR: different gds layers are mapped to the same gds
  layer number:layer1, layer2, shared_layer_number
  ```

  This condition can result if a partial layer mapping is provided in the Calibre query server command file. In general, you should not specify layer mappings (using GDS MAP commands) in the Calibre query server command file.
- Missing pin (x,y) information in the Calibre ideal netlist induces a warning message instructing you to include the relevant Calibre Connectivity Interface query commands in the command file.

  For example, if the Calibre query server command `LAYOUT NETLIST PIN LOCATIONS YES` is not used during the query server run, the StarRC tool issues the following warning:

  ```
  WARNING: Unable to determine terminal locations for "0"
  of device "n". This instance will not be netlisted.
  ```
- Composite `STAMP` commands in the rule file are not supported and result in an SX-1242 warning message. Simple `STAMP` commands are supported.

See Also

- Running the Calibre Query Server

Cross-Referenced Extraction in the Calibre Flow

In the Calibre Connectivity Interface flow, the StarRC tool supports layout-based cross-referencing with the `XREF: YES` command. In this mode, all nets and devices that occur in

Feedback

the ideal layout netlist appear in the parasitic netlist, using schematic net and instance or device names wherever possible. Special naming techniques for handling various merging situations are described in the general discussion of the XREF command.

Schematic and layout cross-referencing in the Calibre Connectivity Interface is based on the layout. This means that the parasitic RC netlist mirrors the structure, connectivity, and quantities of nets, instances, and devices present in the actual layout. Cross-referencing tables are used to annotate schematic names onto these nets, instances, and devices wherever matches are made during the LVS run.

For cross-referenced ideal devices, the model name specified with the Calibre NETLIST MODEL suboption is used with the XREF: YES command, whenever this suboption exists for a particular DEVICE command in the Calibre LVS rule file. Otherwise, the default Calibre model name is used. The StarRC tool always uses the default model name with the XREF:NO command because the layout netlist from the Calibre tool uses that convention.

If a net does not exist in the Calibre NXF file, the StarRC tool uses the layout net name with the prefix specified by the XREF_LAYOUT_NET_PREFIX command (default: ln_). For example, if layout net A did not match anything in the LVS run and does not exist in the Calibre NXF file, the StarRC tool writes ln_A in the parasitic netlist. Similarly, if an instance does not exist in the Calibre IXF file, the StarRC tool uses the layout instance name with the prefix specified by the XREF_LAYOUT_INST_PREFIX command (default: ld_).

Calibre Support of LVS Black Box Flows

You can perform LVS verification using incomplete subcell information by having the LVS tool not compare the functional contents of the subcell. In such cases the cell is called a black box, meaning that the LVS tool treats all instances of the cell as equivalent between schematic and layout without comparing the contents of the cell. The Calibre tool uses the LVS BOX syntax. The only constraint for a black-box cell is that cell ports must correspond between the schematic and layout.

To enable proper StarRC cross-referencing of Calibre LVS BOX cells, you must use the Calibre NET XREF WRITE command to enable cross-referencing using the XREF command in the StarRC tool. The LNXF keyword improves cross-referencing of nets that are not compared during LVS checking.

To cross-reference LVS black-box cells, add the following Calibre query server command to the query command file:

```
NET XREF WRITE file_name LNXF [BOX]
```

Note:

The LNXF construct of the NET XREF WRITE command requires Calibre version 2014.3 or later.

To enable Calibre LVS BOX capability, use the optional keyword `BOX`. Its effects on the NXF file are as follows:

- Within the NXF file, the Calibre tool lists hcells (or equivalence points) for all LVS BOX cells. Equivalence point lines begin with the percent character (%) in the IXF and NXF files. Note that equivalence points for LVS BOX cells are not listed in the NXF file if the keyword BOX is not used in the Calibre Query Server command file.
- Under the hcell (equivalence point) described in the NXF file, the Calibre tool lists all matched ports for the cell. This enables the StarRC tool to cross-reference all ports for LVS BOX cells.

An example of output from the Calibre query server `NET XREF WRITE LNXF BOX` command is as follows:

```
% inv1 4 inv1 6 BOX
0 vss 0 vss
0 vdd 0 vdd
0 b 0 b
0 a 0 a
```

Using this information, the StarRC tool

- Reports cell instances for all LVS BOX cells using schematic instance names, whenever a match for the instance occurs in the Calibre IXF file.
- Reports all port connections to LVS BOX instances using schematic port names, whenever a match for port names occurs in the Calibre NXF file.

Just as with non-LVS BOX cells, the Calibre tool writes interconnect polygon information to the AGF file for LVS BOX cells. You should specify Calibre BOX cells with the `SKIP_CELLS` command in the StarRC command file. If you do not do this, the StarRC tool issues a warning and adds the BOX cell to the StarRC skip cells list. Because these cells must be specified as StarRC skip cells, the StarRC tool treats the contents of LVS BOX cells in a gray-box manner by extracting capacitive interactions from extracted nets to polygons contained in such cells.

Table 11 lists concepts, syntax, and files for the Calibre black box feature.

Table 11 Calibre Black Box Feature

Term	Description
LVS BOX	Calibre rule-file syntax for listing cells whose contents are not to be verified during LVS, but are LVS equivalent.
hcell (or equivalence point)	An expression of LVS equivalence between a schematic cell master and a layout cell master.

Feedback

Table 11 Calibre Black Box Feature (Continued)

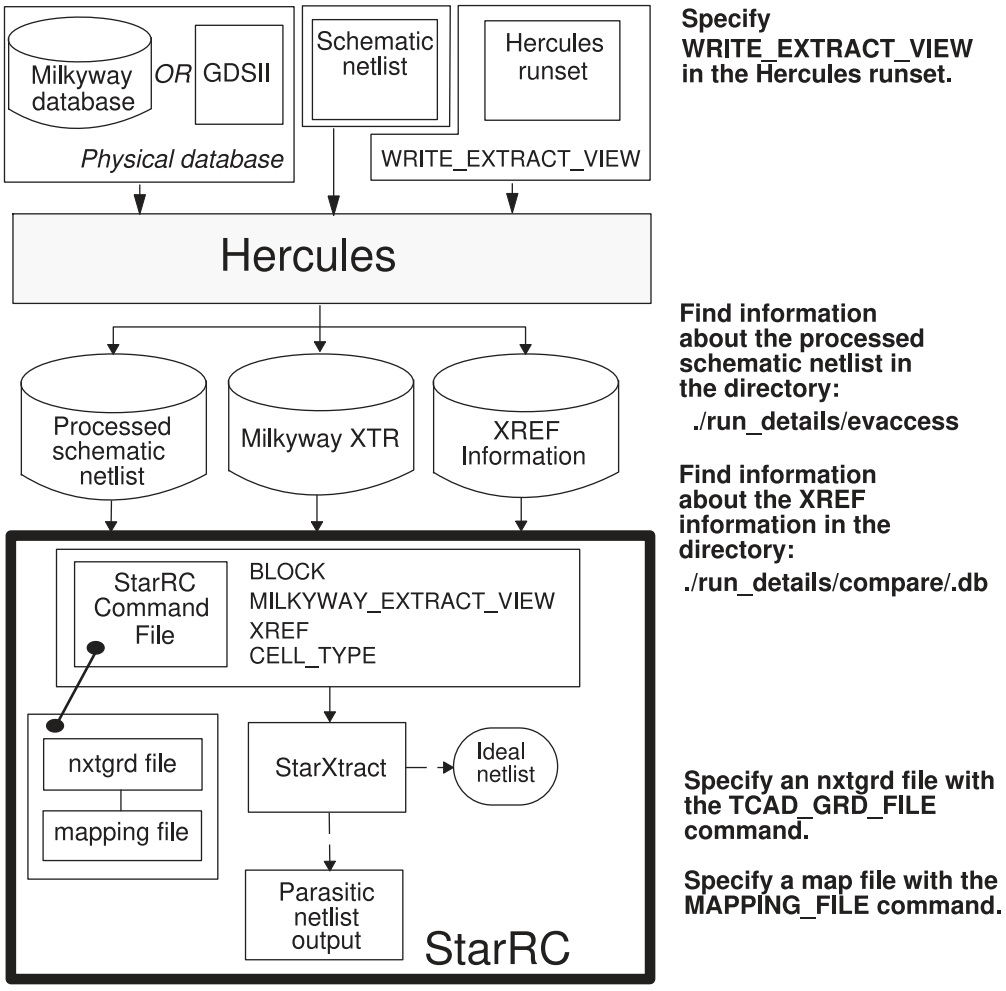
Term	Description
IXF file	Calibre query server output file listing all instance (device and cell) matches between schematic and layout verified during LVS; this file is required by the StarRC tool whenever XREF is activated.
NXF file	Calibre query server output file listing all net matches between schematic and layout verified during LVS; this file is required by the StarRC tool whenever XREF is activated.

Feedback

The Hercules LVS Tool Flow

To create a parasitic netlist output using the Hercules transistor-level extraction flow, you must provide a physical database, schematic netlist, and Hercules runset as shown in Figure 12. To connect the Hercules runset to the StarRC tool, include the `WRITE_EXTRACT_VIEW` Hercules command in the Hercules runset. In the StarRC command file, you must specify the `BLOCK`, `XREF`, `CELL_TYPE`, and `MILKYWAY_EXTRACT_VIEW` commands. Also, the path to the nxtgrd and mapping files must be specified in the StarRC command file. The connected Hercules database, or Milkyway XTR view, can be generated as a parasitic netlist or an ideal extraction of the layout from an original GDSII or Milkyway database. A net in the database that is not annotated with layout text is assigned a numerical net identifier by the Hercules tool.

Figure 12 Hercules Extraction Flow



Feedback

GDSII to XTR View Translation in Hercules Flows

A GDSII file contains no layer connectivity or ideal netlist information. The Hercules tool establishes layer connectivity and extracts an ideal layout netlist using rules defined in the Hercules runset. A connected version of the database is then stored in the form of a Milkyway XTR view database for input to the StarRC tool.

When generating the XTR view, follow these rules:

- The term SUBSTRATE is a reserved keyword in the Hercules runset and cannot be assigned as a `TEMP` or `PERM` layer for any Hercules command.
- In the case of a place-and-route Milkyway database, the Milkyway XTR view generated by Hercules should be written to a unique library because the technology information is different from that of the original library.
- Hercules runsets must be prepared in accordance with StarRC rules for device terminal and connectivity generation.

The following Hercules runset example contains Hercules commands applicable to a cell-level GDS flow for use with the StarRC tool:

```
header{
    layout_path = .
    inlib = library.gds
    format = stream
    block = top
}

 assign_property {
    instance_name (4)
}

assign {
    poly   (1)    text(1;1)
    cont   (2)
    metal1 (3)    text(3;1)
    via1   (4)
    metal2 (5)    text(5;1)
    via2   (6)
    metal3 (7)    text(7;1)
}

text_polygon metal3.text {
     cell_list = { top }
     size = 0.1
     text_list = { IN1 IN2 OUT }
} temp = io_pin

connect {
     poly metal1 by cont
```

```
     metal1 metal2 by via1
     metal2 metal3 by via2
     metal3 by io_pin
}

text {
     poly by poly
     metal1 by metal1
     metal2 by metal2
     metal3 by metal3
}

write_extract_view {
     library_name = TOP
     library_path = .
}
```

You must include a WRITE_EXTRACT_VIEW command as shown in the example to enable Hercules to write out a Milkyway XTR view. It is from this XTR view that the StarRC tool derives the layout net annotation, layout device annotation, and layout connectivity.

The TECHNOLOGY_OPTIONS statement in the Hercules runset can have an impact on extraction runtime. A command-line option for Hercules, -rcxt, sets the TECHNOLOGY_OPTIONS statement for optimal StarRC performance; you should use the -rcxt option for transistor-level or hierarchical designs.

Cross-Referenced Extraction in the Hercules Flow

Multi-equivalence points in the layout are supported for cross-referenced extraction. Multi-equivalence points are points in the layout where one or more layout cells equal to a single schematic cell. The xy coordinates of the instance ports and top-level ports are reported.

If the Hercules IGNORE_CASE=TRUE statement is specified in the runset, all schematic names are uppercase in the StarRC netlist. You must set CASE_SENSITIVE=NO in the command file if IGNORE_CASE=TRUE in the Hercules runset.

Note:

Check that the Hercules runset does not set CREATE_VIEWS=FALSE in the EVACCESS_OPTIONS section. Successful use of the StarRC XREF command requires that this option be either set to TRUE (the default) or left unspecified.

Certain memory designs might encounter errors in the XrefHN step of an XREF-enabled flow. For example,

```
ERROR: StarXtract
ERROR: Found layout instance SRAMblock258x532#448 of equiv
ERROR: SRAMblock258x532#448 which is not a valid equiv point
```

The SRAMblock instances are generated by the Hercules LVS engine when MEMORY_ARRAY_COMPARISON is set to TRUE (this is the default). The SRAMblock instances do not exist in the original schematic or layout netlists.

In general, the MEMORY_ARRAY_COMPARISON variable does not affect the LVS results in most memory designs. For nonmemory designs, you do not need to change this option. The StarRC XREF command options do not support the MEMORY_ARRAY_COMPARISON variable. You should set it to FALSE for memory designs that encounter this error when running LVS.

If the StarRC tool finds an instance that is connected to the net “0” in the schematic netlist, the following error message is issued:

```
Build XrefHN
ERROR: StarXtract
ERROR: Schematic instance "MM15/SRC" is connected to ground "0".
ERROR: To prevent incorrect result, rename net "0" to another name.
Warnings: 0    Errors: 1    (See file xrefhn.sum)
```

To resolve this error, change the net name “0” to a different name in the schematic netlist and rerun Hercules LVS before starting a new StarRC run.

All StarRC XREF command modes support port ordering with the SPICE_SUBCKT_FILE command. The port list should match between the schematic cell definitions and the SPICE_SUBCKT_FILE definitions. The StarRC tool generates net names in the instance port format for ports that are present in the SPICE_SUBCKT_FILE definitions but missing from the schematic or layout to ensure that the port count and order always match the SPICE_SUBCKT_FILE definitions.

When you specify a series merging in Hercules Compare, the StarRC tool reports ln_ for noncross-referenced internal nets.

Placement Information for the HSIM Reliability Flow

An interface to HSIM is provided through a Detailed Standard Parasitic Format (DSPF) file for both hierarchical and flat post-layout simulation flows. The Synopsys HSIM tool can read hierarchical or flat DSPF files to perform hierarchical or flat timing and reliability analysis.

An important output of reliability analysis is a thermal map showing voltage (IR) drop and electromigration violations provided by HSIM. Because the HSIM product is netlist based, the reliability analysis thermal map is generated using node information (*|S, *|I, *|P) provided in the DSPF netlist. In a flat extraction, all nodes are shown with respect to the origin of the top cell and a thermal map can be drawn without ambiguity. However, in a hierarchical flow, each node in a hierarchical cell’s DSPF is shown with respect to its origin. To map these nodes to the next level of hierarchy, you must know the placement of the cell in the next level of hierarchy with rotation and flip attributes.

To generate placement information, specify the following options:

```
SKIP_CELL_PLACEMENT_INFO_FILE: YES
SKIP_CELL_PLACEMENT_INFO_FILE_NAME: file_name
```

When the `PLACEMENT_INFO_FILE` command is set to `YES`, the StarRC tool generates the *blockname*.placement_info file with the following information:

- Angle
- Reflection
- Cell location
- Cell name
- Cross-referenced instance name

The following example shows the output:

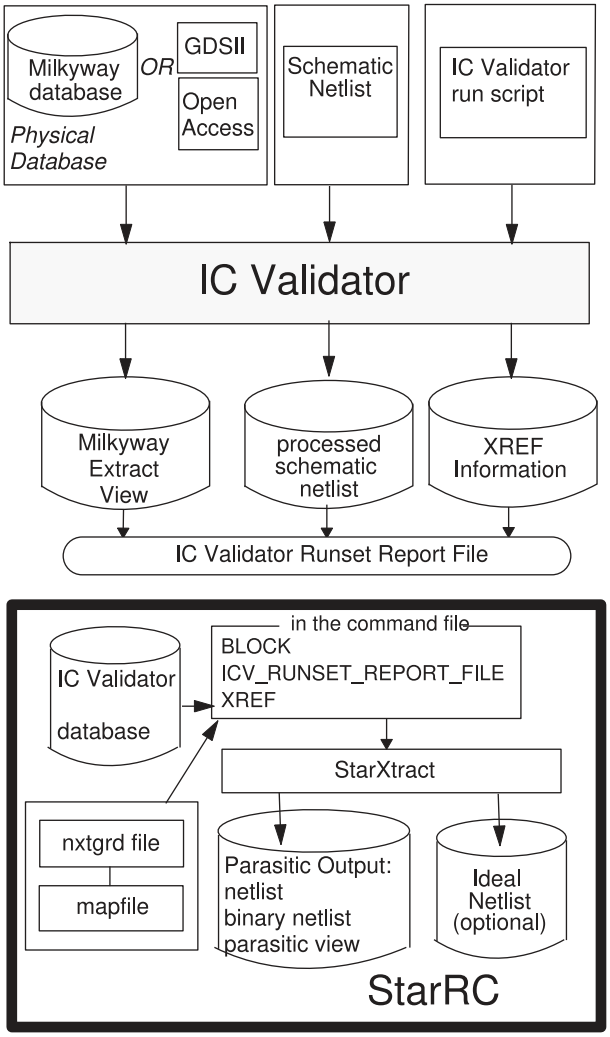
```
* PLACEMENT FILE
* VENDOR "Synopsys, Inc."
* PROGRAM "StarRC X-2005.06"
* DATE "Mon Oct 24 17:48:56 2005"
* UNIT " MICRONS"

TOP_CELL = cell_name
instance_name cell_name x_coord y_coord angle reflection

XSI_0  INV1  49 132   0  NO
XSI_50 XOR2 484 132 180  NO
XSI_61 XOR2 124 312 180 YES
```

Feedback

The IC Validator LVS Tool Flow

To create a parasitic netlist with the IC Validator transistor-level extraction flow, you must provide a physical database, schematic netlist, and an IC Validator runset script to generate an IC Validator database, as shown in Figure 13. The IC Validator database or the IC Validator runset report file can be used as input for the StarRC tool.

Figure 13 IC Validator Flow



1. Specify input data for IC Validator

2. Specify pex_runset_report_file in the runset script.

All IC Validator results or database information is recorded in the runset report file.

Feedback

Steps in the IC Validator Extraction Flow

Follow these steps to use an IC Validator database with the StarRC tool:

- In the IC Validator run script, specify the following command:

  `pex_runset_report_file`

- In the StarRC command file, include these commands:

  `BLOCK`, `XREF`, `CELL_TYPE`, and `ICV_RUNSET_REPORT_FILE`

- (Optional) In the IC Validator run script, specify the runset report file name in the `pex_report_handle` command. By default, the file name is specified as `pex_runset_report` in the $ICV_HOME_DIR/includes/rcxt_public.rh file.

The first two modifications are the required changes to run the IC Validator transistor-level extraction flow. The resulting IC Validator database or IC Validator runset report file can then be used to generate a parasitic netlist or an ideal extraction of the layout from an original GDSII database or Milkyway place and route database.

Examples of Script Files

IC Validator Runset Script

The following example shows the required commands in an IC Validator runset script with the default runset report file name.

```
pex_report_handle = pex_runset_report_file();
pex_generate_results(
    pex_matrix = pex_matrix,
    device_extraction_matrix = my_devices,
    device_db = device_db,
    layout_database = mw_handle,
    pex_process_map_file = pex_process_handle,
    pex_runset_report_file = pex_report_handle
);
```

To change the runset report file name, modify the `pex_report_handle` command specification, which is shown as *my_report* in the following example. All paths listed in the `pex_runset_report_file` command are absolute paths.

```
pex_report_handle = pex_runset_report_file("my_report");
```

StarRC Command File

In the StarRC command file, add the following command:

```
ICV_RUNSET_REPORT_FILE : pex_runset_report
```

Feedback

Cross-Referenced Extraction in the IC Validator Tool

Multi-equivalence points in the layout are supported for cross-referenced extraction. Multi-equivalence points are points in the layout where one or more layout cells equal to a single schematic cell. The xy coordinates of instance ports and top-level ports are reported.

The IC Validator tool supports a selective case-sensitive operation. The StarRC `CASE_SENSITIVE` command might not cover all IC Validator case sensitivity combinations.

Specify the `pex_generate_results` function in the IC Validator runset script to run the IC Validator and StarRC flow, as shown in the following example.

```
pex_generate_results(
    pex_matrix = pex_matrix,
    device_extraction_matrix = my_devices,
    device_db = device_db,
    layout_database = mw_handle,
    pex_process_map_file = pex_process_handle,
    pex_runset_report_file = pex_report_handle
);
```

All StarRC cross-referencing modes support port ordering with the `SPICE_SUBCKT_FILE` command. The port count and port order in the schematic cell definitions and the `SPICE_SUBCKT_FILE` subcircuit definitions should always match. The StarRC tool generates net names in the instance port format for ports that are present in the `SPICE_SUBCKT_FILE` section but missing in the schematic or layout, depending on the options set in the `XREF` command.

Error Conditions

Memory Circuits

Certain memory designs might encounter errors in a cross-referenced flow. For example,

```
ERROR: StarXtract
ERROR: Found layout instance SRAMblock258x532#448 of equiv
ERROR: SRAMblock258x532#448 which is not a valid equiv point
```

The blocks named SRAMblock are generated by the IC Validator tool when the `memory_array_compare` variable is set to `TRUE` (the default). Those blocks do not exist in the original schematic or layout netlists.

In general, the `memory_array_compare` variable does not affect the LVS results in most memory designs. For nonmemory designs, you do not need to change this option. The StarRC `XREF` command options do not support the `memory_array_compare` variable. You should set it to `FALSE` for memory designs that encounter this error when running LVS.

Feedback

Ground Nets

If the StarRC tool finds an instance that is connected to net 0 in the schematic netlist, the following error message is issued:

```
Build XrefHN
ERROR: StarXtract
ERROR: Schematic instance "MM15/SRC" is connected to ground "0".
ERROR: To prevent incorrect result, rename net "0" to another name.
Warnings: 0    Errors: 1    (See file xrefhn.sum)
```

In this case, you must change the net name “0” to a different name in the schematic netlist and rerun LVS before starting a new StarRC run.

Cross-Referencing in Transistor-Level Flows

The StarRC `XREF` command can be used with the IC Validator, Hercules, and Calibre flows.

Table 12 lists definitions for terms related to cross-referencing.

Table 12 Cross-Referencing Terms

Term	Definition
XREF (also known as back-annotation or cross-referencing)	Generation of parasitic netlist containing layout parasitics with schematic-based net and instance names.
Schematic-based XREF	Generation of parasitic netlist containing all devices and nets that occur in a schematic netlist used for LVS. Specified in the StarRC tool with the `XREF:COMPLETE` command.
Layout-based XREF	Generation of parasitic netlist containing all devices and nets that occur in the extracted layout netlist used for LVS. Specified in the StarRC tool with the `XREF:YES` command.
Device merging	Series or parallel combinations of multiple devices by the LVS tool for single electrically equivalent devices. Often necessary to successfully match schematic devices to corresponding layout devices that were implemented as electrically equivalent series or parallel combinations of devices.
Composite device	Resulting device created by the LVS tool when individual layout or schematic devices are merged into series or parallel combinations. When devices are merged, only composite devices participate in the actual layout-to-schematic LVS matching.

Feedback

This section contains the following topics:

XREF: NO

When the `XREF` command is set to `NO`, the StarRC tool reports the layout net names generated during ideal layout extraction. These layout names are either generated or derived from the application of text. The layout cell instance names can either be the original GDSII instance name if the `ASSIGN_PROPERTY` statement in Hercules is used or they can be names generated by Hercules.

XREF: YES

When the `XREF` command is set to `YES`, the StarRC tool does not use name prefixes for merged devices. The tool generates names that correspond to the unmerged schematic names, making simulation easier and improving schematic-based simulation accuracy.

The `XREF:YES` command is layout-based; every layout device and net is reported. If the LVS tool successfully matches layout nets and devices to schematic nets and devices, the StarRC tool uses the schematic names in the parasitic netlist.

When the LVS tool establishes a one-to-one match between schematic net or device and layout net or device names, the StarRC tool writes the layout nets and devices using their matching schematic names.

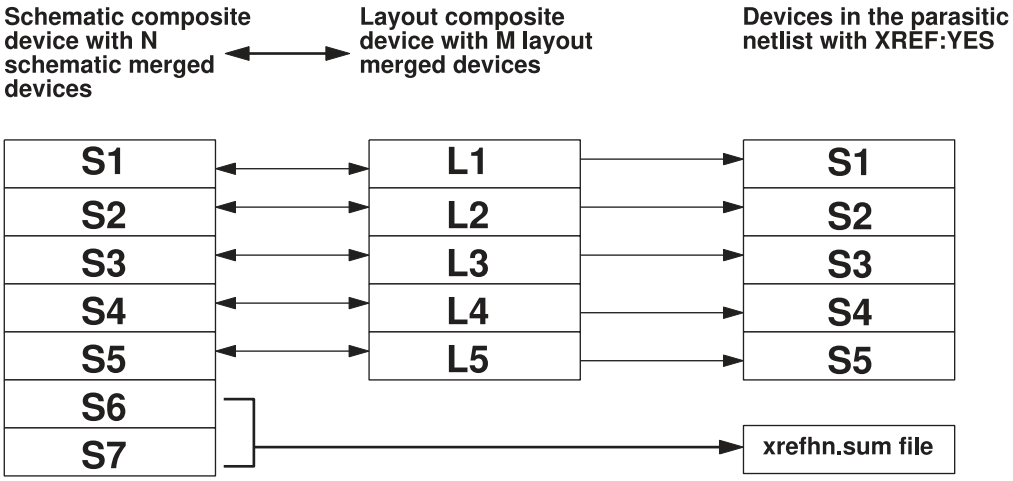
When the LVS tool creates a composite merged device on the schematic side consisting of N merged devices, and matches it to a composite merged device on the layout side consisting of M merged devices, the following rules govern the device names in the netlist generated by the `XREF:YES` command:

- M individual devices are written in the parasitic netlist, corresponding to the M layout devices.
- A one-to-one correspondence is established between the first X devices where X is the smaller of N or M within the schematic composite device and the layout composite device. The first X devices are written in the parasitic netlist using their corresponding schematic device names.

Feedback

- If the number of schematic devices exceeds the number of layout devices (N>M), the remaining (N-M) schematic devices are not referenced in the schematic netlist, as shown in Figure 14.

Figure 14 Merged Device Handling: More Schematic Devices than Layout Devices



Feedback

- If the number of layout devices exceeds the number of schematic devices (N<M), the remaining (M-N) layout devices are mapped back to the top of the schematic device list, appended by @[number] characters, as shown in Figure 15.

Figure 15 Merged Device Handling: Fewer Schematic Devices than Layout Devices

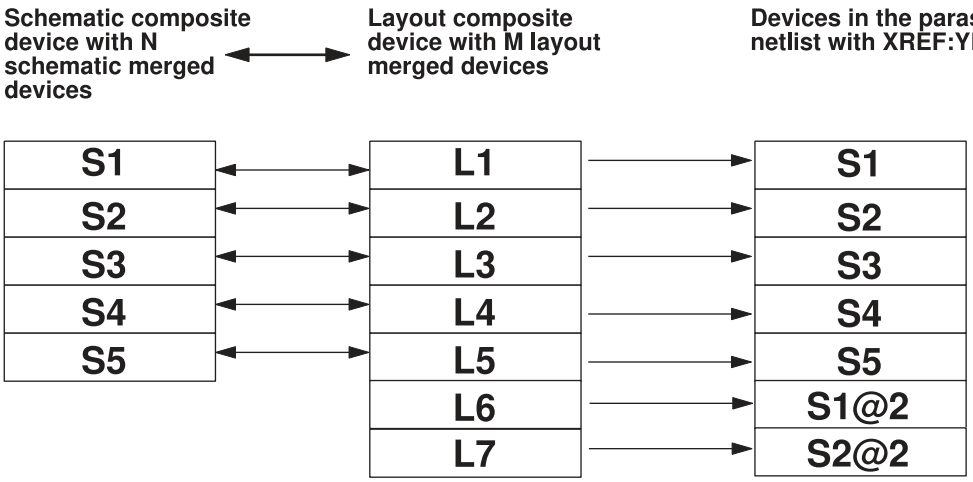


Table 13 shows the device naming conventions for devices participating in composite N:M merged devices with the `XREF:YES` setting. If a layout net is generated within a merged composite device, its name in the netlist contains the layout net name with a prefix specified by the `XREF_LAYOUT_NET_PREFIX` command.

Table 13 XREF:YES Device Naming Conventions

No. of devices (schematic : layout)	Device instance names	XREF-info report file
1:1	S1	
1:N	S1 S1@2 S1@3 … S1@N	
N:N	S1 S2 S3 … SN	

Feedback

Table 13 XREF:YES Device Naming Conventions (Continued)

No. of devices (schematic : layout)	Device instance names	XREF-info report file
N:M (N>M)	S1 S2 S3 … SM	alias S(M+1) S1 alias S(M+2) S2 …
N:M (N<M)	S1 S2 S3 … SN S1@2 S2@2 S3@2 … SN@2 …	
N:1	S1	alias S2 S1 alias S3 S1 … alias SN S1
0:M (filtered schematic devices)	<XREF_LAYOUT_INST_PREFIX>L1	

XREF: COMPLETE

When the `XREF` command is set to `COMPLETE`, the StarRC tool reports every skip cell and device in the schematic. Parasitics appear in the netlist only for nets that have been successfully cross-referenced to schematic nets. Nets that do not cross-reference to a schematic net are treated as ideal connections. Schematic model names are reported for skip cells and primitive devices.

Internal nets of the SERIES or PATHS merged devices do not have *|NET sections when the `XREF: COMPLETE` command is used; the netlist result is ideally included in the Instance Section.

Table 14 describes how the StarRC tool computes properties for the Hercules and IC Validator flows.

Table 14 Device Property Reporting

Schematic: Layout	Width	Length	AD, AS, PD, PS
MERGE_PARALLEL			

Feedback

Table 14 Device Property Reporting (Continued)

Schematic: Layout	Width	Length	AD, AS, PD, PS
1:y	Summation of all the width values of the n MOS devices in the layout.	Smallest length values out of all of the layout MOS devices.	Summation of all the AD, AS. PD. and PS values of the NMOS devices in the layout.
x:1	Width of the layout MOS divided by x for each device.	Same length of the layout MOS device for each device.	AD, AS. PD. and PS values of the layout MOS device divided by x for each device
x:x	Corresponding width value from the layout. No calculation is performed.	Corresponding length value from the layout.	Corresponding AD, AS. PD. and PS values from the layout.
x:y	Summation of all the width values of the MOS devices in the layout, divided by x for each device.	Smallest length value of all of the layout MOS devices.	Summation of all the AD, AS. PD. and PS values of the MOS devices in the layout, divided by x for each device.
MERGE_SERIES			
x:x	Corresponding width value from layout. No calculation is performed.	Corresponding length value from layout. No calculation is performed.	Corresponding AD, AS. PD. and PS values from layout. No calculation is performed.
MERGE_SERIES_GATE			
1:y	Average width of the MOS devices in the layout.	Summation of all the length values of the MOS devices in the layout.	Summation of all the AD, AS. PD. and PS values of the MOS devices in the layout.
x:1	Same width as the layout MOS device for each device.	Length of the layout MOS divided by x for each device.	AD, AS. PD. and PS values of the layout MOS device divided by x for each device.
MERGE_PATHS			
MERGE_PATHS is a mixture of the MERGE_PARALLEL, MERGE_SERIES, and MERGE_SERIES_GATE cases. If there is a multiple-layout-cell-to-single-schematic-cell equivalency in the LVS, the StarRC tool randomly chooses only one of the layout cells and uses the layout properties of that cell in when generating the netlist of all the XREF cells.			

Feedback

Cross-Referencing By Device Property

The StarRC and IC Validator tools provide a feature to improve the accuracy of cross-referencing parallel merged devices when the number of schematic devices is different from the number of layout devices.

The IC Validator tool can map devices into groups based on transistor width and provide this information in the cdb file. The feature is controlled by the following IC Validator environment variables:

```
XREF_PARALLEL_MAP
XREF_PARALLEL_MAP_MEMBER_LIMIT
XREF_PARALLEL_MAP_ITER_LIMIT
```

The StarRC tool uses the information in the cdb file automatically. The `XREF` command must be set to `YES`.

For example, assume the schematic parallel composite device contains these transistors:

```
M100 w=0.6u
M101 w=0.5u
```

The layout parallel composite device contains these transistors:

```
M1 w=0.3u
M2 w=0.3u
M3 w=0.5u
```

The final StarRC output (using the `XREF: YES` command) is as follows:

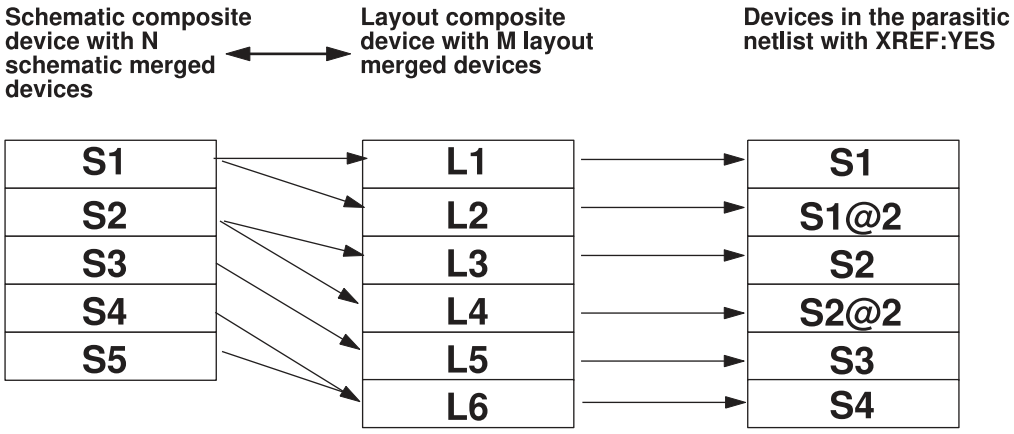
```
M1 -> M100
M2 -> M100@2
M3 -> M101
```

The combined width of the parallel-merged layout devices M1 and M2 is 0.6 um, equal to the width of schematic device M100. Layout device M3 is the same as schematic device M101.

Figure 16 illustrates naming for scenarios where one schematic device maps to more than one layout device (S1 mapping to L1 and L2) and where more than one schematic device maps to one layout device (S4 and S5 mapping to L6).

Feedback

Figure 16 Merge By Device Property



Feedback

How the XREF Command Affects SPF Netlists

The following examples show the effect of `XREF` command settings on an SPF netlist.

XREF: NO

```
*|NET I_0/N_33 1.49e-02PF
*|I (I_0/I_39/M2:SRC I_0/I_39/M2 SRC B 0 -402.5 36.25)
*|I (I_0/I_39/M1:SRC I_0/I_39/M1 SRC B 0 -402.5 11)
*|I (I_0/I_41/M3:GATE I_0/I_41/M3 GATE I 1e-15 -419.5 36.25)
*|I (I_0/I_41/M1:GATE I_0/I_41/M1 GATE I 1e-15 -417 11)
Cg9 I_0/I_39/M2:SRC 0 4.82142e-15
Cg10 I_0/I_39/M1:SRC 0 6.20537e-15
Cg11 I_0/I_41/M3:GATE 0 1.62791e-15
Cg12 I_0/I_41/M1:GATE 0 2.25542e-15
R5 I_0/I_39/M2:SRC I_0/I_41/M3:GATE 141.086
R6 I_0/I_39/M2:SRC I_0/I_39/M1:SRC 1.41411
R7 I_0/I_39/M2:SRC I_0/I_41/M1:GATE 66.6508
R8 I_0/I_39/M1:SRC I_0/I_41/M3:GATE 95.2203
R9 I_0/I_39/M1:SRC I_0/I_41/M1:GATE 44.9831
R10 I_0/I_41/M3:GATE I_0/I_41/M1:GATE 625.714
```

XREF: YES

```
*|NET B6 0.0223556PF
*|I (MM18@2:g MM18@2 g I 0.00425085 12.725 143.652)
*|I (MM18:g MM18 g I 0.00425085 11.875 143.652)
*|I (MM17@2:s MM17@2 s B 0 23.565 143.652)
*|I (MM17:s MM17@2 s B 0 23.565 143.652)
Cg30 MM18@2:g 0 2.0949e-15
Cg31 MM18:g 0 2.75462e-15
Cg32 MM17@2:s 0 2.03411e-15
Cg33 MM17:s 0 1.87562e-15
Cg34 B6:79 0 1.57134e-15
Cg35 B6:85 0 7.22334e-15
R7537 MM17:s B6:177 32.2773
R7538 MM17@2:s B6:173 48.3553
R7539 MM18:g B6:85 32.0359
R7540 MM18@2:g B6:79 32.2773
R7541 B6:85 B6:79 32.0359
```

XREF: COMPLETE

```
*|NET x0/n33 1.49e-02PF
*|I (x0/x39/M2:SRC x0/x39/M2 SRC B 0 -402.5 36.25)
*|I (x0/x39/M1:SRC x0/x39/M1 SRC B 0 -402.5 11)
*|I (x0/x41/M3:GATE x0/x41/M3 GATE I 1e-15 -419.5 36.25)
*|I (x0/x41/M1:GATE x0/x41/M1 GATE I 1e-15 -417 11)
Cg1 x0/x39/M2:SRC 0 4.82142e-15
Cg2 x0/x39/M1:SRC 0 6.20537e-15
Cg3 x0/x41/M3:GATE 0 1.62791e-15
```

```
Cg4 x0/x41/M1:GATE 0 2.25542e-15
R1 x0/x39/M2:SRC x0/x41/M3:GATE 141.086
R2 x0/x39/M2:SRC x0/x39/M1:SRC 1.41411
R3 x0/x39/M2:SRC x0/x41/M1:GATE 66.6508
R4 x0/x39/M1:SRC x0/x41/M3:GATE 95.2203
R5 x0/x39/M1:SRC x0/x41/M1:GATE 44.9831
R6 x0/x41/M3:GATE x0/x41/M1:GATE 625.714
```

Feedback

Parameterized Cells

A parameterized cell (PCell or container cell) is placed in the layout to represent a device. The cell contents are sized during placement to achieve certain geometries and functionality in the device. For simulation and analysis purposes, the entire contents of the cell are often characterized and modeled as a unit, including all conductor effects inside the cell boundary.

How StarRC Layer-Based Rules Affect Parameterized Cells

By default, the StarRC tool defines the boundary between interconnect parasitic effects and intradevice effects through layer-based rules for each of the following device types:

- Capacitance – These rules allow the tool to discard all capacitance considered to be inside the device by ignoring capacitance between certain predetermined device terminal layer combinations.
- Resistance – You can include or exclude parasitic resistance on a layer-by-layer basis. However, because a parameterized cell is typically characterized as a unit, it is simpler and more accurate to use the cell boundary to separate intradevice parasitic effects that are discarded from interconnect effects, but retained in the netlist.

The StarRC tool uses gray box extraction techniques to handle parameterized cells, which means that the cell boundary is used to delineate the device boundary and knowledge of the layers within the cell is not required. This extraction method allows you to extract PCell devices without the need for device layer manipulation in the extraction rule file.

How LVS Tools Handle Parameterized Cells

During ideal device extraction and layout versus schematic (LVS) checking, you can use device extraction commands to extract one or more logical devices from polygons inside a parameterized cell. The ideal layout netlist output then contains a hierarchy level that represents the PCell or container cell. Inside this container cell is the extracted device, which represents the design at the lowest level of hierarchy (the transistor level).

Conversely, the schematic netlist might or might not contain a level of cell hierarchy corresponding to the container cell. Possible scenarios are as follows:

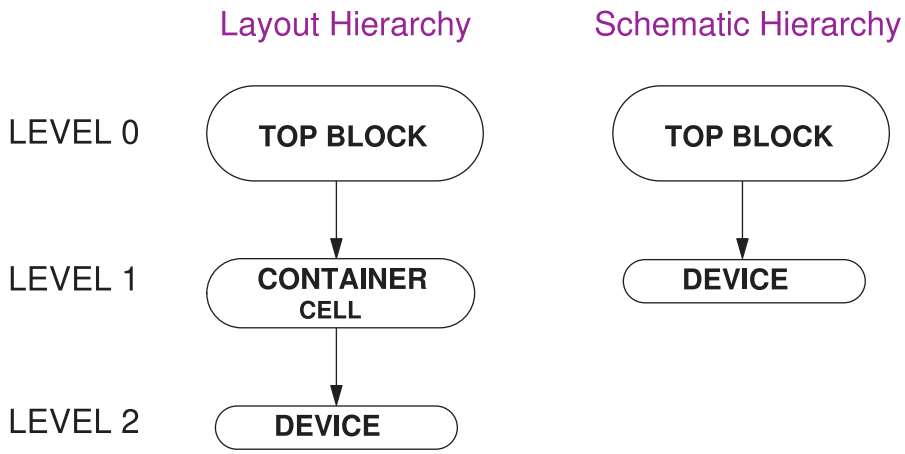
- Single Layout Device With No Container Cell
- Multiple Layout Devices With No Container Cell
- Layout Devices in a Schematic Container Cell

Feedback

Single Layout Device With No Container Cell

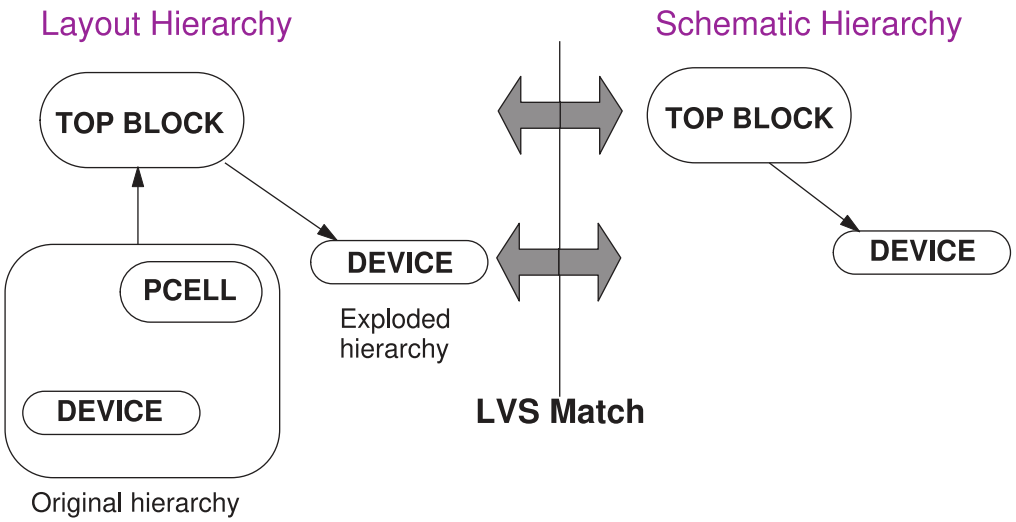
The schematic netlist might contain only the device instance, not the container cell instance, because the parameterized cell is a physical object, not a logical object. This creates a nonuniform hierarchy between the layout and schematic, because the layout has an extra level of cell hierarchy not present in the schematic as shown in Figure 17. To match the layout and schematic, the layout versus schematic tool expands (explodes) the layout container cell in the layout netlist so that the extracted device appears in the top block of the layout netlist. This results in an equal hierarchy between the layout and schematic for complete LVS matching, as shown in Figure 18.

Figure 17 PCell Layout and Schematic Hierarchy; Single Device Inside Container



Feedback

Figure 18 LVS Matching of PCell Devices via Layout PCell Explosion



Multiple Layout Devices With No Container Cell

The schematic netlist might contain only the device instance, but not the schematic container cell instance. In addition, the layout container cell might contain multiple instances of device primitives, which might or might not be of the same device type. For example, a parameterized cell representing a MOSFET might have well diodes extracted inside the layout container cell in addition to the MOS primitive itself, as shown in Figure 19.

In this case, LVS must explode the layout container cell, as shown in Figure 20, because no corresponding cell hierarchy exists in the schematic. All primitives originally inside the layout container cell are then matched by LVS processing to primitives in the schematic.

Feedback

Figure 19 Multiple Devices Inside Layout Container Cell

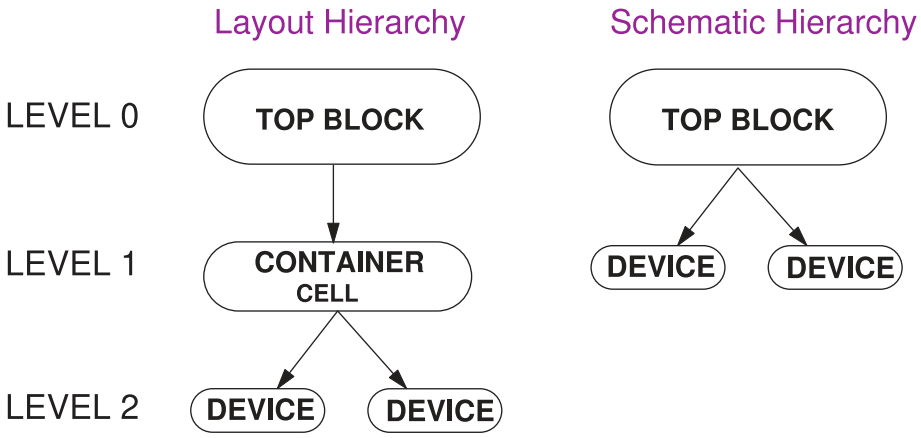
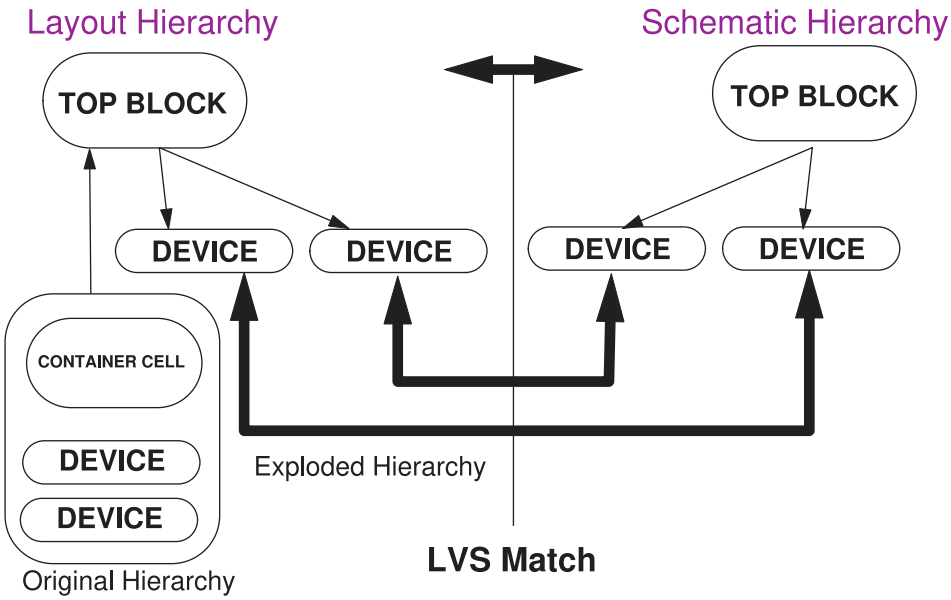


Figure 20 Multiple Devices Inside Layout Container Cell, After Exploding the Container Cell

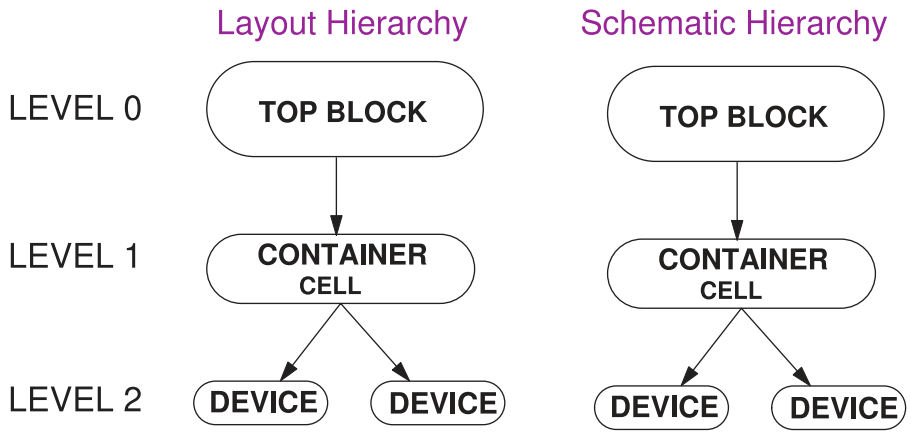


Layout Devices in a Schematic Container Cell

In this scenario, the schematic netlist might contain a device instance inside a schematic cell that has an EQUIV point (in the Hercules or IC Validator tools) or HCELL (in the Calibre tool) corresponding to the layout container cell. The layout container cell might contain one or more instances. In this case, LVS maintains the equivalent schematic and layout hierarchy to match the schematic device to the layout device as shown in Figure 21.

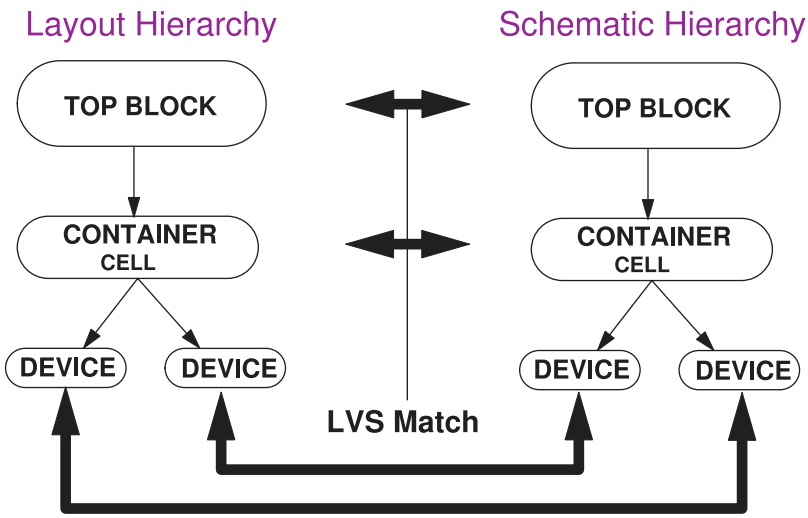
Feedback

Figure 21 PCell Layout and Schematic Hierarchy: Container Cell in Schematic



In a design that has equivalence between the schematic and layout, no explosion is needed as shown in Figure 22. In this case, LVS maintains the equivalent schematic and layout hierarchy to match the schematic device to the layout device.

Figure 22 LVS Matching of PCell Layout and Schematic Hierarchy



Extracting Parameterized Cells

To extract a parameterized cell (PCell) as a fully characterized gray box cell unit during parasitic extraction, specify the `SKIP_CELLS` command in the StarRC command file. The following functions change for handling parameterized cells.

Feedback

Gray Box Handling

Parasitic resistance is extracted up to the instance port location for each cell port. Capacitive interactions between top-level nets and the material inside the cell are extracted as ground capacitances in accordance with the StarRC standard gray box extraction method. Port capacitance is not included in the total capacitance for the net connecting to the port.

The IGNORE_CAPACITANCE Command

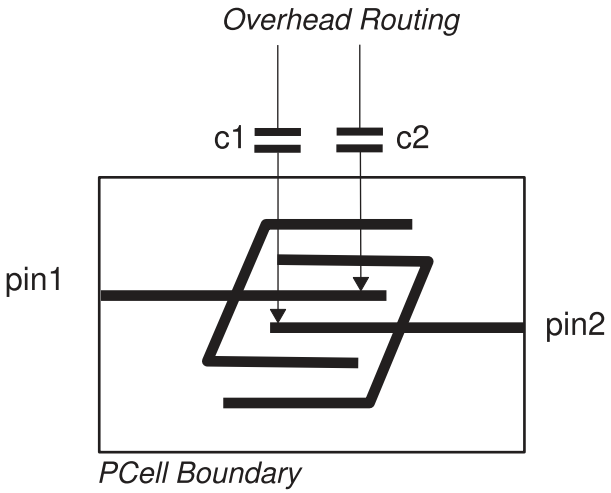
During extraction, the parameterized cell is treated as a gray-box skip cell. Functions related to the `IGNORE_CAPACITANCE` command are disabled for skip cells (but not for non-PCell devices) because layer-based capacitance removal is not required.

Extracting Coupling Capacitances

The StarRC tool reports coupling capacitances for two additional conditions related to PCell structures. You can report coupling capacitances between overhead nets and PCell pins and coupling capacitances between different PCell pins reported in the generated netlist. You can do this by specifying the `COUPLE_TO_PCELL_PINS` command.

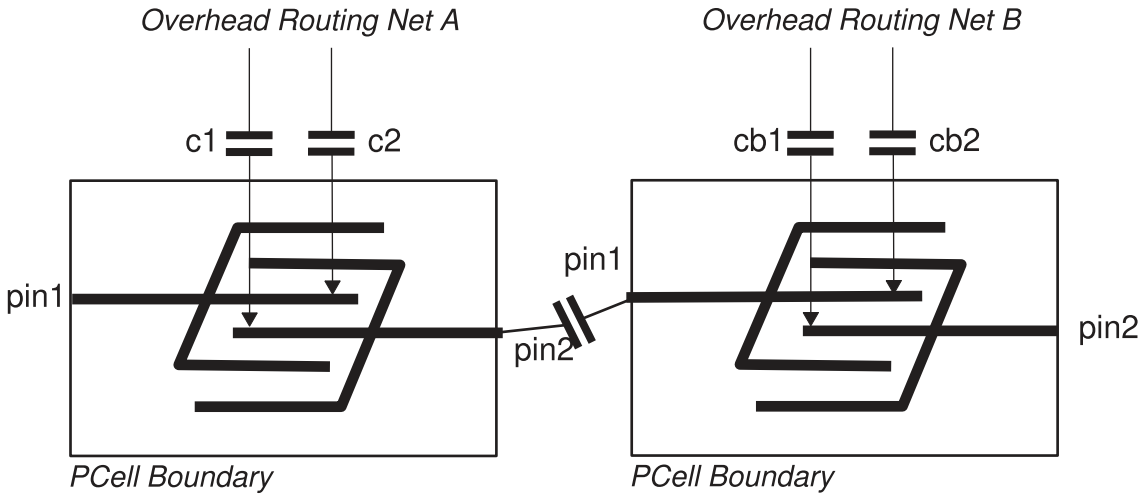
With this command, the coupling capacitances between PCell pins and overhead nets are reported or grounded, depending on the command. Figure 23 shows the extraction of overhead nets to PCell pins. Figure 24 shows the extraction of overhead nets.

Figure 23 Extraction of Overhead Nets



Feedback

Figure 24 Extraction of Overhead Net



Retaining Coupling Capacitance Between Top and Skip Cell Levels

You can use the `COUPLE_NONCRITICAL_NETS` command to retain coupling capacitances between top-level parent routing and skip cell child net routing, where the fully routed child (DEF or CEL view) routing net names are used for coupling node names. This feature exists for the Milkyway flow using the SPEF netlist format.

To specify which noncritical nets are to be retained with an added prefix, use the `COUPLE_NONCRITICAL_NETS` and `COUPLE_NONCRITICAL_NETS_PREFIX` commands. Use the `COUPLE_NONCRITICAL_NETS_SUBNODE_SUFFIX` command to add a subnode suffix to the noncritical nets. Use the `NONCRITICAL_COUPLING_REPORT_FILE` command to specify an output file containing all the capacitances that are coupled to the noncritical nets.

SKIP_PCELLS Netlist Behavior

The entire logical content of a PCell appears in the netlist, as follows:

- All devices inside the PCell container cell are represented in the DSPF instance section.
- All `*|I` lines in the DSPF file represent connections to individual devices inside the container cell.

The logical content of the DSPF netlist (devices in the instance section or `*|I` lines) are identical to a generated netlist if a `SKIP_PCELLS` operation has not been performed on the

Feedback

PCell container. This method supports different LVS configurations of PCells. The netlist result is as follows:

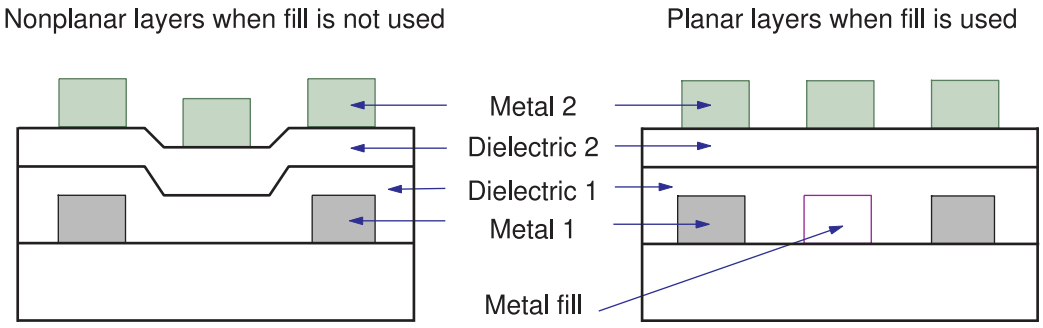
- The devices inside the PCell container cell are generated with the same device names that are used when `SKIP_PCELLS` operations are not performed.
- All geometric device properties for the devices inside the container cell are written as normal in the DSPF instance section.
- If the PCell container cell has a port that is not connected to a device inside the container cell, that port is ignored during netlist output.
- Any internal nodes inside the PCell container cell (for example, nodes that are not instance ports of the container cell) are treated as ideal nets in the DSPF netlist.
- All specified `INSTANCE_PORT` commands for PCells are automatically set to `SUPERCONDUCTIVE`.

Feedback

Metal Fill

Metal fill is commonly included in designs that are manufactured with chemical-mechanical polishing (CMP) steps or conformal dielectric layers. These processes cause characteristic patterns of top surface height nonuniformity. Figure 25 shows how the presence or absence of a metal line in an underlying layer affects the planarity of the top layer. To improve the planarity of the top layer and thereby improve process yield, metal fill polygons are inserted into the design.

Figure 25 Effect of Metal Fill on Layer Planarity



Metal fill can be grounded or floating. Grounded metal fill is connected to power or ground nets by vias. Floating metal fill has no connection to signal, power, or ground nets. Both types of fill might exist in the same layout.

You can model metal fill in several ways. For more information, see the following topics:

- Emulated Metal Fill

  A simple estimation of fill effects, used only in the early stages of the place-and-route flow

- Virtual Metal Fill

  A more realistic estimation of fill effects

- Real Metal Fill

  Accurate extraction for signoff based on real metal fill polygons, which are read either directly from the design database or from a separate GDSII or OASIS® file

See Also

- The Metal Fill Reuse Flow

Feedback

Emulated Metal Fill

Emulated metal fill is a method of modeling metal fill effects in the early design stage when the metal fill is not yet available for the design. You can save the fill emulation parameters in a TLUPlus file for use in a place-and-route or synthesis tool. Actual fill design data is not required.

Caution:

The StarRC tool ignores emulated fill parameters if defined in the ITF file.

Fill emulation is not intended for use in a signoff flow because emulated fill does not represent the actual fill in a design.

TLUPlus models are used by the parasitic extractor in other Synopsys tools, including the Fusion Compiler, IC Compiler II, and Design Compiler Topographical Mode tools. TLUPlus models are not used by the StarRC tool.

Model emulated metal fill as follows:

1. Create an ITF file that includes the `FILL_TYPE`, `FILL_SPACING`, `FILL_RATIO`, and `FILL_WIDTH` options for a conductor layer.
2. Use the grdgenxo tool to create a TLUPlus file from the ITF file.
3. Use the TLUPlus file in a place-and-route tool at the early design stage or in a synthesis tool before the handoff to a place-and-route tool.

The ITF options are as follows:

- The `FILL_WIDTH` option specifies the average size of the fill features, as shown in Figure 26. This value primarily affects lateral capacitance.
- The `FILL_SPACING` option specifies the spacing between the metal fill region and the signal lines, as shown in Figure 26. This value primarily affects lateral capacitance.
- The `FILL_RATIO` option specifies the utilization rate of the available metal tracks, as shown in Figure 27. The maximum value is 1.0, which means that all available metal tracks are occupied with metal features. This value primarily affects vertical capacitance (capacitance between vertical metal layers).
- The `FILL_TYPE` option specifies whether the emulated fill is grounded or floating.

For emulated fill modeling, the `FILL_WIDTH`, `FILL_SPACING`, and `FILL_RATIO` options must all be specified. Empty space in the conductor layer is modeled as though it were filled with metal of the same layer. You can specify whether it should be floating or grounded by using the `FILL_TYPE` option; if you omit this option, the default is grounded.

Feedback

Figure 26 FILL_SPACING and FILL_WIDTH Commands for Emulated Metal Fill

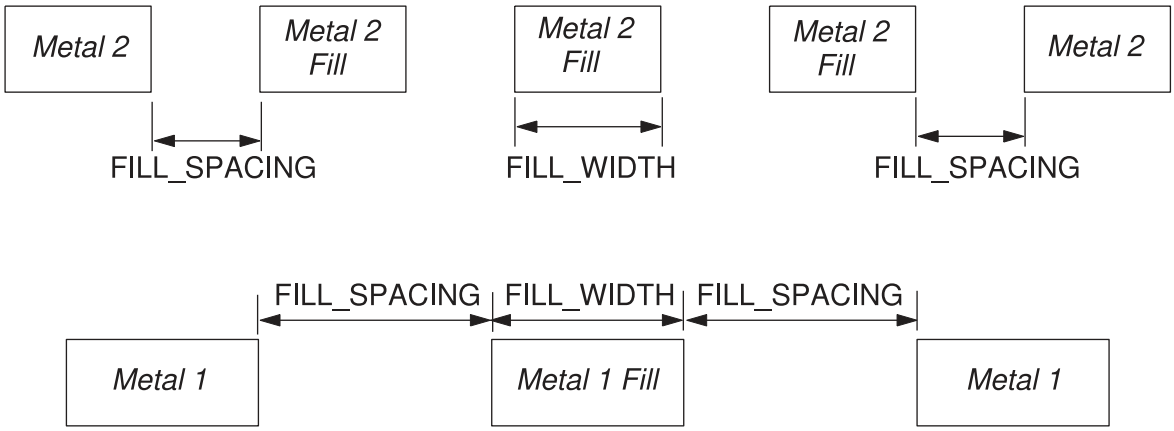
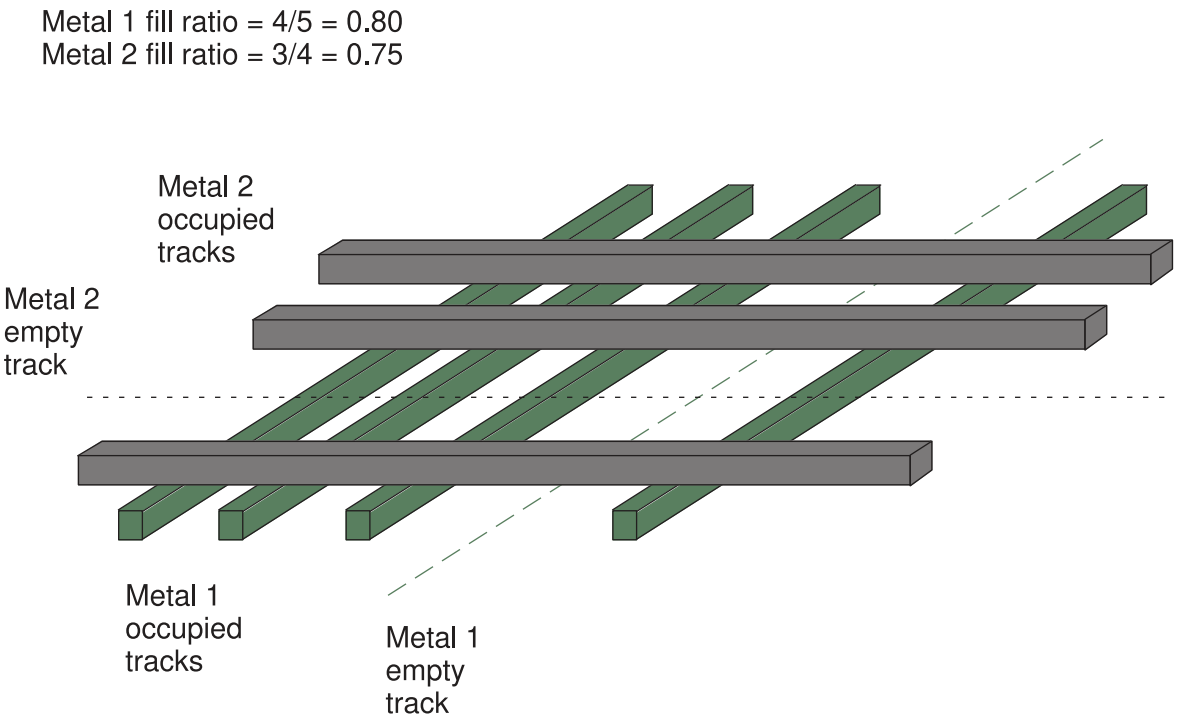


Figure 27 FILL_RATIO Command for Emulated Metal Fill



Feedback

Virtual Metal Fill

Before inserting of real metal fill, you can insert virtual metal fill polygons for better estimation of parasitics and timing early in the design process. Virtual metal fill emulates the metal fill that exists directly on parasitic components.

Caution:

Virtual metal fill is not intended for use in a signoff flow because it does not represent the actual fill in a design.

The following StarRC commands control virtual metal fill insertion:

- `VIRTUAL_METAL_FILL_PARAMETER_FILE`: Specifies a file that contains parameters to define the metal fill.
- `VIRTUAL_METAL_FILL_OPTIONS_FILE`: Specifies a file with complex design-specific parameters and rules to achieve better correlation with real metal fill characteristics.
- `VIRTUAL_METAL_FILL_POLYGON_HANDLING`: Specifies how to treat the metal fill.
- `VIRTUAL_METAL_FILL_NDR_NETS`: Specifies a list of nondefault rule nets for special handling.
- `VIRTUAL_METAL_FILL_PARAMETERIZE`: Allows you to create a parameter file by analyzing a design with real metal fill using the same design technology.
- `VIRTUAL_METAL_FILL_EXCLUDED_CELLS`: Excludes cells from virtual metal fill placement.

When the StarRC tool inserts virtual metal fill, it ensures that the design remains DRC clean by honoring minimum widths and spacings. The tool adjusts the values provided in the virtual metal fill parameter file, as follows:

- If there is not enough space to insert virtual metal fill based on the provided parameters, the tool does not insert any fill.
- If the spacing between signal lines is not large enough to accommodate an integer number of tracks, the tool increases the fill-to-fill spacing to fit the maximum number of fill tracks between signal lines.
- If necessary, the tool also increases the fill-to-signal spacing.

You can use virtual metal fill in combination with real metal fill. Spacings to real metal fill polygons are the same as spacings to virtual metal fill polygons. The `VIRTUAL_METAL_FILL_POLYGON_HANDLING` command applies to virtual metal fill placement on designs with real metal fill. The `TRANSLATE_FLOATING_AS_FILL` command is honored for real fill polygons when the flow contains both real fill and virtual fill.

To exclude cells from virtual metal fill placement, list the cells in the `VIRTUAL_METAL_FILL_EXCLUDED_CELLS` command. If you specify a cell for exclusion, you can optionally include the `fill_blockage_excluded_cells` parameter in the parameter file to specify the distance of virtual metal fill polygons from the excluded cells.

The Virtual Metal Fill Parameter File

The StarRC tool accepts the following file types for the virtual metal fill parameter file:

- A metal fill parameter .rh file generated by the IC Validator tool describes track fill placement. This file is recognized by the presence of the layer mapping definition in the header. You can specify multiple .rh files in the argument list of the `VIRTUAL_METAL_FILL_PARAMETER_FILE` command. In such a case, the first file in the list must contain the layer mapping information.
- A file generated by the Fusion Compiler or IC Compiler II tool with the `set_extraction_options -virtual_metalfill_parameter_file` command. You can specify only one of these files in the argument list.
- A file generated by the StarRC tool by setting the `VIRTUAL_METAL_FILL_PARAMETERIZE` command to `YES`. Use this command to analyze real metal fill and write a parameter file based on its layout. You can analyze a metal fill design strategy one time, then use the generated parameter file for subsequent extraction runs for the same technology.
- A manually-created ASCII file. You can specify only one of these files in the argument list.

The StarRC tool enforces parameter uniformity for design layers that are mapped to the same ITF layer. For example, if 5 design layers are mapped to the same ITF layer, but the parameter file only contains parameters for 3 of those layers, the tool assigns parameters to the 2 missing layers. The tool uses the parameters from the first design layers assigned to that ITF layer.

A manually-created ASCII file must contain one line for each database layer for which to create virtual metal fill.

To skip parameter generation for a design layer, provide a line in the parameter file that contains only the layer name and the keyword `S` for routing direction.

Otherwise, each line must use the following syntax:

```
layer_name direction
           fill_width min_fill_length     max_fill_length      \
           min_fill_route_w_spacing       fill_route_l_spacing \
           min_fill_fill_w_spacing        fill_fill_l_spacing  \
           fill_pwr_w_spacing             fill_pwr_l_spacing   \
           min_fill_blockage_w_spacing    fill_blockage_w_spacing  \
           min_fill_ndr_w_spacing         fill_ndr_l_spacing    \
```

Feedback

```
fill_chip_w_spacing            fill_chip_l_spacing    \
fill_blockage_excluded_cells
```

The `min_fill_ndr_w_spacing` and `fill_ndr_l_spacing` parameters are used only for nondefault rule (NDR) nets specified with the `VIRTUAL_METAL_FILL_NDR_NETS` command.

The StarRC tool checks the validity of the parameters and issues warning or error messages as needed. Table 15 describes the parameters. The last 7 parameters have defaults. You can omit the last 1, 3, 5, or 7 parameters at the end of a line to use the defaults. Some parameters must be specified in pairs, as noted in the table.

Note:

All dimensions in the virtual metal fill parameter file must be in nanometers. Dimensions should be the scaled size if a half-node scale factor is in use.

Table 15 Virtual Metal Fill Parameter File Values

Parameter	Type	Valid values	Description
`layer_name`	string	Any valid database layer name	A layer for which to create virtual me tal fill
`direction`	character	`V` (vertical) `H` (horizontal) `U` (unknown) `S` (skip)	Routing direction of virtual metal fill shapes
`fill_width`	float	Greater than or equal to the WMIN value for the layer	Width of the virtual metal fill shapes in the direction perpendicular to the routing direction
`min_fill_length`	float	Greater than or equal to the WMIN value for the layer	Minimum length of the virtual metal fill shapes in the direction parallel to the routing direction
`max_fill_length`	float	Greater than or equal to the min_fill_length value	Maximum length of the virtual metal fill shapes in the direction parallel to the routing direction
`min_fill_route_ w_spacing`	float	Greater than or equal to the SMIN value for the layer	Minimum spacing between virtual fill shapes and design shapes in the direction perpendicular to the routing direction
`fill_route_ l_spacing`	float	Greater than or equal to the SMIN value for the layer	Spacing between virtual fill shapes and design shapes in the direction parallel to the routing direction

Feedback

Table 15 Virtual Metal Fill Parameter File Values (Continued)

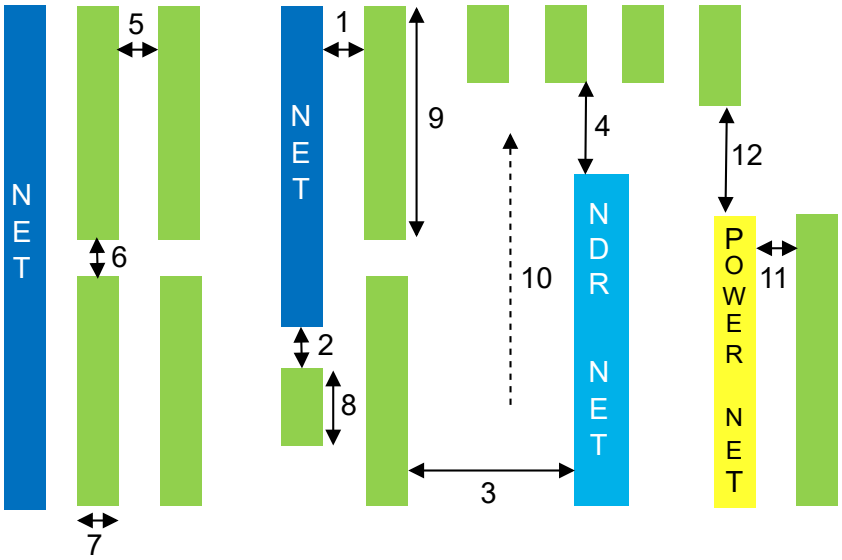
Parameter	Type	Valid values	Description
`min_fill_fill_ w_ spacing`	float	Greater than or equal to the SMIN value for the layer	Minimum spacing between virtual fill shapes in the direction perpendicular to the routing direction
`fill_fill_ l_spacing`	float	Greater than or equal to the SMIN value for the layer	Spacing between virtual fill shapes in the direction parallel to the routing direction
`fill_pwr_w_spac ing`	float	Greater than or equal to the SMIN value for the layer	Spacing between virtual fill shapes with power net and design shapes in the horizontal direction If you do not specify this parameter, the tool uses the value specified with the `fill_route_w_spacing` parameter.
`fill_pwr_l_spac ing`	float	Greater than or equal to the SMIN value for the layer	Spacing between virtual fill shapes with power net and design shapes in the vertical direction If you do not specify this parameter, the tool uses the value specified with the `fill_route_l_spacing` parameter.
`min_fill_blocka ge_ w_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `min_fill_route_w_spacing`. Must specify along with `fill_blockage_l_spacing`.	Minimum spacing between virtual fill shapes and design blockage shapes in the direction perpendicular to the routing direction
`fill_blockage_ l_ spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `fill_route_l_spacing`. Must specify along with `min_fill_blockage_w_spaci ng`.	Spacing between virtual fill shapes and design blockage shapes in the direction parallel to the routing direction
`min_fill_ndr_ w_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `min_fill_route_w_spacing`. Must specify along with `fill_ndr_l_spacing`.	Minimum spacing between virtual fill shapes and NDR nets in the direction perpendicular to the routing direction

Feedback

Table 15 Virtual Metal Fill Parameter File Values (Continued)

Parameter	Type	Valid values	Description
`fill_ndr_ l_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `fill_route_l_spacing`. Must specify along with `min_fill_ndr_w_spacing`.	Spacing between virtual fill shapes and NDR nets in the direction parallel to the routing direction
`fill_chip_ w_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to 0. Must specify along with `fill_chip_l_spacing`.	Minimum spacing between virtual fill shapes and the chip boundary in the direction perpendicular to the routing direction
`fill_chip_ l_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `fill_route_l_spacing`. Must specify along with `fill_chip_w_spacing`.	Spacing between virtual fill shapes and the chip boundary in the direction parallel to the routing direction
`fill_blockage_ excluded_cells`	float	Greater than or equal to the SMIN value for the layer. Defaults to 200 nm.	Spacing between virtual fill shapes and excluded cells

Figure 28 Virtual Metal Fill Parameters



1. min_fill_route_w_spacing
2. fill_route_l_spacing
3. min_fill_ndr_w_spacing
4. fill_ndr_l_spacing
5. min_fill_fill_w_spacing
6. fill_fill_l_spacing
7. fill_width
8. min_fill_length
9. max_fill_length
10. direction (V)
11. fill_pwr_w_ spacing
12. fill_pwr_l_ spacing

Feedback

Nondefault Rule Net Handling

The `min_fill_ndr_w_spacing` and `fill_ndr_l_spacing` parameters are used only for nondefault rule (NDR) nets specified with the `VIRTUAL_METAL_FILL_NDR_NETS` command.

If NDR rules are defined in the design database, and those spacings are larger than the `min_fill_ndr_w_spacing` and `fill_ndr_l_spacing` parameters, the rules in the design database take priority.

An NDM format design created with the Fusion Compiler or IC Compiler II tool might also contain virtual shielding rules. If the `INDESIGN_VIRTUAL_SHIELDING` command is set to `YES` in the StarRC command file, the StarRC tool honors the virtual shielding rules when determining how to place virtual metal fill polygons.

Some nets might have multiple sets of NDR rules defined. For example, the `VIRTUAL_METAL_FILL_NDR_NETS` command might specify nets that already have NDR rules in the design database. In this case, the tool uses the largest spacing from all rules when inserting metal fill polygons near an NDR net.

Feedback

Real Metal Fill

You can provide metal fill information from any combination of the following sources:

- The design database (Fusion Compiler, IC Compiler II, Milkyway, or LEF/DEF designs)
- One or more GDSII files, by using the METAL_FILL_GDS_FILE command

  In this case, you must also provide layer mapping information by using the GDS_LAYER_MAP_FILE command.
- One or more OASIS files, by using the METAL_FILL_OASIS_FILE command

  In this case, you must also provide layer mapping information by using the OASIS_LAYER_MAP_FILE command.

Shapes in a GDSII or OASIS file are treated as metal fill objects for extraction if the following conditions are met. All other data in the file is ignored.

- The shapes are on a layer that is listed in the respective mapping file.
- The shapes are referenced by the top-level block definition in the metal fill file or by a child cell of the top-level block.
- The top-level block name of the metal fill data matches the top-level block name of the design database. You can change the top-level block name for the metal fill data by using the METAL_FILL_BLOCK_NAME or METAL_FILL_GDS_BLOCK commands.

The tool translates metal fill polygons into an internal format before performing extraction. Translation depends on the setting of the METAL_FILL_POLYGON_HANDLING command or on the fill handling options in the mapping file specified by the GDS_LAYER_MAP_FILE command (or for OASIS files, the OASIS_LAYER_MAP_FILE command).

The METAL_FILL_POLYGON_HANDLING command treats metal fill features as follows:

- IGNORE (the default)

  Does not translate metal fill, even if it is present in the database.
- FLOATING

  Processes all metal fill as floating, even if the fill is placed as grounded fill.

  In this mode, capacitance is calculated between signal and fill polygons and between fill polygons. After extraction, the StarRC tool reduces fill nodes on the fly and calculates the equivalent capacitance between signal nets and the capacitance to ground for signal nets.

- GROUNDED

  Processes all metal fill as grounded, even if the fill is placed as floating fill.

- AUTOMATIC

  Processes design database fills based on the section in which they appear.

You can process a combination of floating and grounded metal fill. Treat grounded fill shapes as part of the power network by using text to identify them. Alternatively, you can use the METAL_FILL_GDS_FILE_NET_NAME command (or for OASIS files, the METAL_FILL_OASIS_FILE_NET_NAME command) to name specific nets to be grounded.

The 3-D field solver can handle floating metal fill. The StarRC tool automatically prepares the field solver input based on the METAL_FILL_POLYGON_HANDLING command and the layer handling specified in the layer mapping file.

Coupling Capacitance on Floating Metal Fill

If you use the METAL_FILL_POLYGON_HANDLING: FLOATING or REMOVE_FLOATING_NETS: YES command, floating fill polygons are not generated in the parasitic netlist. Floating fill nets do not have a *D_NET (SPEF format) or *|NET (SPF format) section in the netlist, and the floating nets have no coupling capacitance to other nets.

When you use the REMOVE_FLOATING_NETS:YES command, the StarRC tool determines the coupling capacitance from signal nets to floating nets and adds this coupling capacitance to the total ground capacitance of the signal net.

If the METAL_FILL_POLYGON_HANDLING command is set to FLOATING and floating metal features are designated as fill, the StarRC tool extracts coupling capacitance to floating fill polygons but does not include the floating fill in the parasitic netlist.

Instead of grounding the coupling capacitors to fill polygons, the StarRC tool performs reduction on the capacitors that connect to fill nodes. The tool computes equivalent capacitances and eliminates the fill nodes. The effects of this approach are as follows:

- If nets couple to each other through fill polygons, the netlist has a coupling capacitor between these two nets when the METAL_FILL_POLYGON_HANDLING command is set to FLOATING. When the REMOVE_FLOATING_NETS command is set to YES, the coupling capacitance to the floating nets appears as additional ground capacitance.
- Nets that normally do not couple to each other might couple to each other after fill is added to the design. When the METAL_FILL_POLYGON_HANDLING command is set to FLOATING, a coupling capacitor between these nets appears in the netlist. This increases the accuracy of signal integrity analysis because crosstalk effects induced by metal fill can be considered.

Feedback

Specifying Metal Fill in the Design Database

You can denote metal fill in a design in the following ways:

- Fusion Compiler or IC Compiler II (NDM format) designs

  The StarRC tool accepts metal fill polygons in the FILL view for a block.

- IC Compiler (Milkyway format) designs

  The StarRC tool accepts metal fill polygons in the FILL, CEL, or FRAM view for a block.

- LEF/DEF designs

  LEF/DEF versions 5.4 and later support two forms of syntax for specifying metal fill. Floating metal fill polygons are specified in the FILLS section of the DEF file. If the fill polygons are tied to power and ground nets, they are specified in the SPECIALNETS section (part of special wiring with SHAPE defined as FILLWIRE) for the power and ground nets.

- GDSII and OASIS files

  The StarRC tool can read metal fill polygons from a separate GDSII or OASIS file. For this flow, the design database can be in NDM, Milkyway, LEF/DEF, or GDSII format (a Milkyway XTR view generated by the Hercules or IC Validator tools or an AGF file generated by the Calibre or IC Validator tools). The GDSII or OASIS file must contain only metal fill polygons, because all the polygons from the file are considered to be fill.

  The handling mode for metal fill imported from a file can be specified on either a global or layer-specific basis.

  No floating fill should exist in the design XTR view because the StarRC tool cannot automatically identify these fills. You can attach VSS text to identify grounded fills in the physical layout and make them a part of the existing ground network.

  You can specify a flat GDSII (or OASIS) file by using the `METAL_FILL_GDS_FILE` (or `METAL_FILL_OASIS_FILE`) command. If you would like to attach all fills to a specific net, use the `METAL_FILL_GDS_FILE_NET_NAME` (or `METAL_FILL_OASIS_FILE_NET_NAME`) command.

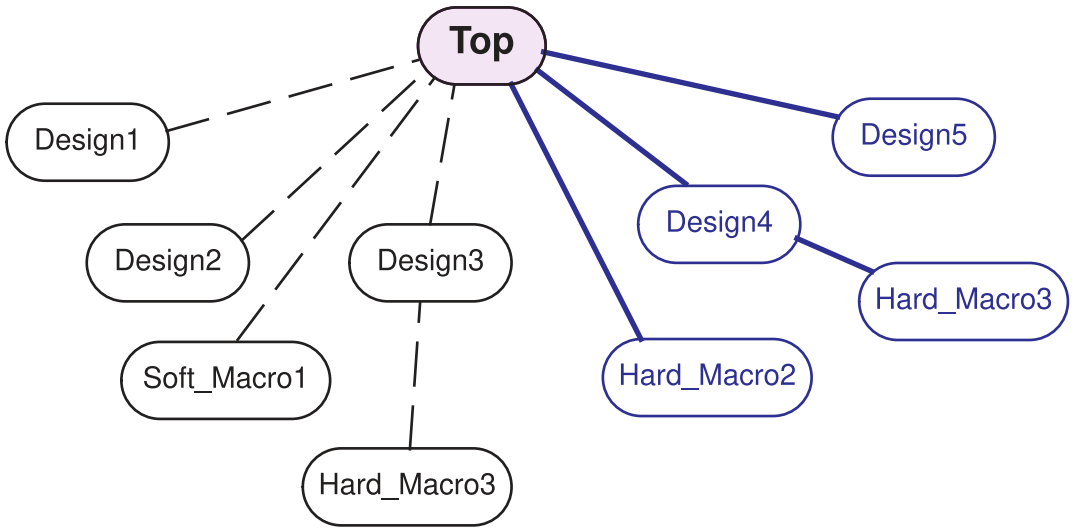
  To ensure that the fill structures are identified properly in the database, use the `GDS_LAYER_MAP_FILE` (or `OASIS_LAYER_MAP_FILE`) command.

Feedback

Metal Fill For Hierarchical Designs

A hierarchical design is composed of a top cell that references multiple child cells (design macros), each of which might reference other design macros, soft macros, or hard macros, as illustrated in Figure 29. A design macro might either contain fill information directly or rely on a fill cell (fill macro) to provide the fill information.

Figure 29 Example Hierarchical Design



Fill information can also be provided hierarchically, with a top-level fill macro that references multiple child fill macros. By default, the StarRC tool processes the design hierarchy and the fill hierarchy independently. With careful attention to detail, using hierarchical design macros and hierarchical fill macros together can be successful.

To provide more control over metal fill insertion, you can define fill macros to be used with specific design macros by creating a mapping file and specifying it with the `METAL_FILL_BLOCK_MAPPING_FILE` command. Each line of the mapping file contains a design macro name followed by a fill macro name. You do not have to map all the macros in the design.

Note:

This feature is valid only for LEF/DEF designs and gate-level flows.

For example, a metal fill mapping file for the design shown in Figure 29 might be as follows:

```
Hard_Macro2 Hard_Macro2_fill
Hard_Macro3 Hard_Macro3_fill
Design5 Design5_fill
Design4 Design4_fill
```

In this example, design macros Hard_Macro2, Hard_Macro3, Design4, and Design5 are targeted for hierarchical fill. The metal fill mapping file specifies the fill macros to be used with each of these design macros.

The StarRC tool handles the metal fill for the remaining design macros (such as Design1 and Soft_Macro1) using the default method, which follows the hierarchy information provided in the metal fill top cell.

The following usage notes apply:

- A fill macro named in the metal fill mapping file should not be referenced in its parent design's fill cell.

  In this example, fill cell Design4_fill should not reference fill cell Hard_Macro3_fill, because the mapping file controls the associations. The StarRC tool issues a warning message if this improper reference is detected.

  However, a fill cell for Design3 can contain a reference for fill cell Hard_Macro3_fill, because the fill for Design3 and its child cells is not controlled by the mapping file.

- The top-level fill macro should not contain references to child design fill macros that are named in the mapping file.

Reporting Metal Fill

To determine how many polygons are read from different layers, examine the metal fill statistics in the summary files located in the star/summary directory. Metal fill reporting is off by default; to enable it, use the `REPORT_METAL_FILL_STATISTICS: YES` command.

Differentiating Metal Fills

To differentiate metal fills from other design information in the GDS or Oasis layer map file, use the `IP_FILL` keyword. When you specify the `IP_FILL` keyword with the `GDS_LAYER_MAP_FILE` or `OASIS_LAYER_MAP_FILE` command for a specific layer, polygons of the specified layer are considered for metal fill and processed with the `METAL_FILL_POLYGON_HANDLING` command that uses metal fill input files in the GDSII or OASIS format.

To use the `IP_FILL` keyword with the `GDS_LAYER_MAP_FILE` or `OASIS_LAYER_MAP_FILE` command when you use the `METAL_FILL_POLYGON_HANDLING` command, see the syntax of GDS_LAYER_MAP_FILE and OASIS_LAYER_MAP_FILE.

Metal fills from the GDS or OASIS layer map file are applied with the same mask shift, similar to other design information.

In the GDS or Oasis layer map file, when you specify both route and metal fill polygons for a skip cell and if metal fill polygons and interconnect polygons of a skip cell are segregated by a layer and a datatype, the tool treats the metal fill polygons as floating or grounded.

Feedback

This is based on the setting of the IP_FILL keyword in GDS_LAYER_MAP_FILE or OASIS_LAYER_MAP_FILE.

Table 16 shows how the tool treats the skip cell based on settings of the METAL_FILL_POLYGON_HANDLING command with the use of the IP_FILL keyword in GDS or OASIS layer map file.

Table 16 Setting of IP_FILL in GDS or Oasis Layer Map File

METAL_FILL_ POLYGON_HANDLING	GDS_LAYER_MAP_FILE OASIS_LAYER_MAP_FILE		Example of GDS or OASIS layer map file	How the Tool Treats Skip Cell?
	Layer; Datatype	Keyword		
FLOATING	10;11	None, floating, grounded	**<*db layer*> 10 11 FLOATING**	Treats as grounded
AUTOMATIC	10;11	None, floating, grounded	**<*db layer*> 10 11 FLOATING**	Treats as grounded
FLOATING	10;11	IP_FILL	**<*db layer*> 10 11 IP_FILL**	Treats as floating
AUTOMATIC	10;11	IP_FILL	**<*db layer*> 10 11 FLOATING IP_FILL**	Treats as floating
GROUNDED	10;11	IP_FILL	**<*db layer*> 10 11 IP_FILL**	Treats as grounded
AUTOMATIC	10;11	IP_FILL	**<*db layer*> 10 11 GROUNDED IP_FILL**	Treats as grounded
IGNORE	10;11	IP_FILL	**<*db layer*> 10 11 IP_FILL**	Ignores
AUTOMATIC	10;11	IP_FILL	**<*db layer*> 10 11 IGNORE IP_FILL**	Ignores

See Also

- METAL_FILL_POLYGON_HANDLING

The Metal Fill Reuse Flow

The StarRC tool provides a method to improve runtime in an ECO loop by reusing metal fill. This feature is available for gate-level flows.

Feedback

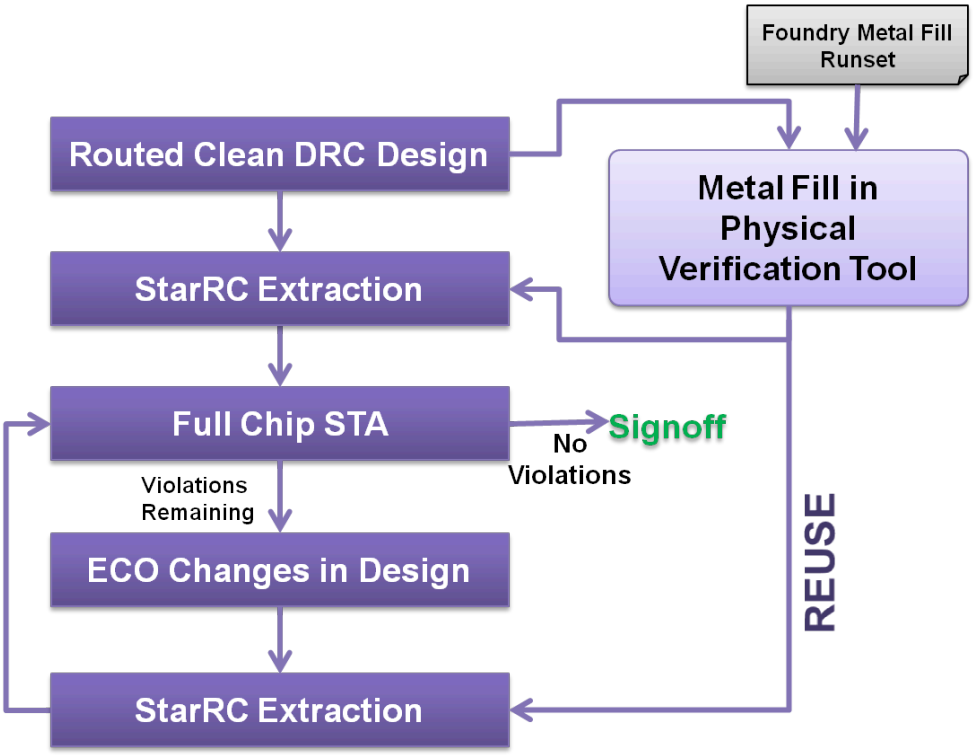
For timing closure, iterative design and analysis runs are frequently necessary to identify and fix problems. The design changes are known as engineering change orders (ECOs). ECO changes might include localized gate placement and sizing changes, net rerouting, and net additions or deletions. After the design is changed, parasitic extraction and timing analysis must be repeated to verify that the issues are resolved.

Metal fill insertion can be a significant part of the overall turnaround time of each ECO loop, even though only a small part of the design might have changed. The metal fill reuse flow provides a way to save time in the ECO loop by reusing metal fill features.

Figure 30 shows the metal fill reuse flow. At the beginning of the design process, metal fill is generated for use in the first StarRC extraction run. After timing analysis, ECO changes might be necessary to fix timing violations. The modified design requires another extraction run, but now the original metal fill design is used, thereby saving runtime.

However, some of the old metal fill structures might short to the new design changes. The StarRC tool resolves these shorts by moving metal fill polygons away from signal nets.

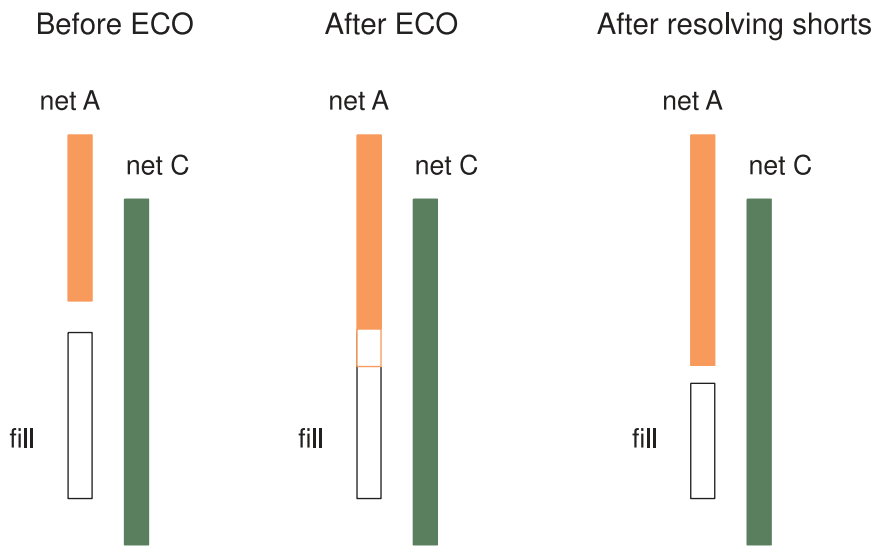
Figure 30 Metal Fill Reuse Flow



The metal reuse flow has very little impact on accuracy because most of the original metal fill is the same throughout the signoff loop. Any metal fill structures that are moved to resolve shorts are moved in a manner consistent with the original fill polygon creation method.

Feedback

Figure 31 Resolving Shorts in the Metal Fill Reuse Flow



The extraction runtime might increase by a small amount, but in most cases the decrease in turnaround time in the overall signoff loop is much larger.

To use the metal reuse flow, follow this procedure:

1. Perform a standard extraction on a design containing metal fill.
2. Perform timing analysis and fix timing violations.
3. Include the following command in the StarRC command file:

   ```
   REMOVE_METAL_FILL_OVERLAP: YES
   ```

4. (Optional) Specify the spacing to use when the StarRC tool resolves shorts between metal fill polygons and signal nets by inserting the value in the `conducting_layers` section of the mapping file as follows:

   ```
   conducting_layers
        overlap_fill_spacing=new_space
   ```

   If you do not specify an explicit spacing value, the StarRC tool uses the `SMIN` value for the conductor layer in the ITF file.

5. Perform another extraction run.
6. Perform another timing analysis and fix timing violations.

7. Repeat the extraction, timing analysis, and design modification steps as needed.
8. If the number of ECO changes is large or the number of metal fill reuse iterations is large, generate new metal fill before proceeding.
9. For final signoff, perform full metal fill insertion followed by full-chip extraction and timing analysis to verify that there are no timing violations.

4

StarRC Extraction and Output

This chapter describes fundamental aspects of the StarRC extraction process and the available reports.

For more information, see the following topics:

- Simultaneous Multicorner Extraction
- The Parasitic Database or GPD
- Directories and Files
- Output Netlists
- Extraction For Electromigration Analysis

Feedback

Simultaneous Multicorner Extraction

Simultaneous multicorner (SMC) extraction optimizes the efficient extraction of multiple process and temperature corners for a design. The simultaneous multicorner flow

- Runs a single extraction operation on a user-defined set of nxtgrd files and operating temperatures
- Generates topologically equivalent individual corner netlists
- Is enabled by default for both gate-level and transistor-level flows
- Can analyze up to 15 process corners, where a process corner is a unique combination of nxtgrd files and pattern density specification

All extraction modes are supported, with the exception of inductance analysis.

For gate-level flows, the StarRC tool automatically detects process and design conditions for which SMC extraction would provide better runtime than single-corner extraction. If the StarRC commands are not set up to allow SMC extraction, the tool issues a message to recommend that you use SMC extraction.

Table 17 summarizes the effect of StarRC command settings on the extraction mode.

Table 17 Effect of Command File Settings on SMC Extraction

SIMULTANEOUS_MULTI_CORNER command	CORNERS_FILE and SELECTED_CORNERS commands	Extraction
YES	present	SMC
not present	present	SMC
NO	not present	single-corner
not present	not present	single-corner
YES	not present (either or both)	error
NO	present (either or both)	error

Feedback

Setting Up Simultaneous Multicorner Extraction

To use the simultaneous multicorner flow, follow this general procedure:

1. Create a corners file that includes all of the defined corners (unique combinations of operating temperatures and nxtgrd files). The maximum number of corners is 15. The corners file must include a `TCAD_GRD_FILE` command for each corner.
2. Add the `CORNERS_FILE` command to the command file to specify the corners file name.
3. Include the `SIMULTANEOUS_MULTI_CORNER: YES` command in the command file (or leave out the command, because the default is `YES`).
4. Add the `SELECTED_CORNERS` command to the command file to specify the list of corners to be extracted. The selected corners must be a subset of the corners defined in the corners file.
5. (Optional) Use the `NETLIST_SMC_FORMULA` command to create a single netlist containing RC values written as formulas that use the corner names as variables. This option is valid only for transistor-level extraction.
6. (Optional) Use the `EXTRACTION: FSCOMPARE` or `FS_EXTRACT_NETS` commands to use the field solver flow during the extraction.
7. (Optional) Use the `DENSITY_OUTSIDE_BLOCK` command to specify pattern density outside the block. This capability requires the Ultra+ license.

The Corners File

A unique corner is a combination of the nxtgrd file and the pattern density specification. The `CORNERS_FILE` command specifies a file that defines all corners that are available for SMC extraction.

The following commands are valid in the corners file for all design databases:

```
CORNER_NAME: name_of_corner
TCAD_GRD_FILE: path_to_nxtgrd_file
OPERATING_TEMPERATURE: temperature_in_Celsius

(optional) MAPPING_FILE: map_file
(optional) VIA_COVERAGE_OPTION_FILE: via_file
(optional) DENSITY_OUTSIDE_BLOCK: density_value
```

For simultaneous multicorner extraction, the StarRC tool uses only the `TCAD_GRD_FILE` and `OPERATING_TEMPERATURE` commands within the corners file. The tool ignores other instances of these commands in the command file.

The CORNERS_FILE and STAR_DIRECTORY commands must follow these naming conventions:

- If the STAR_DIRECTORY command specifies a relative path, you can use either a relative path or an absolute path in the CORNERS_FILE command. For example:

```
STAR_DIRECTORY: star_work
CORNERS_FILE: smc_config
```

- If the STAR_DIRECTORY command specifies an absolute path, you must use an absolute path in the CORNERS_FILE command. For example:

```
STAR_DIRECTORY: /tmp/star
CORNERS_FILE: /remote/.../work_directory/smc_config
```

You can optionally define some aspects of the extraction separately for each corner: the mapping file, the via coverage option file, and RC scaling factors.

When you use the SIMULTANEOUS_MULTI_CORNER: YES command, the nxtgrd files should be specified with the CORNERS_FILE command only. However, if you specify the nxtgrd file in both the corners and StarRC command files, the StarRC tool issues an error message.

Mapping Files for Corners

You can specify a mapping file for each corner by including MAPPING_FILE commands in the corner definitions. If a corner definition does not specify a mapping file, a global mapping file must be specified in the command file. If every corner includes a mapping file, a global mapping file is not necessary; if one exists, it is ignored.

All mapping files for simultaneous multicorner flows must be consistent in terms of the number of database and ITF file layers and their mapping relationships. Each mapping file category, such as conductors or vias, should also be consistent. The only allowed variations are the RPSQ value for conductors and the RPV value for vias.

Via Coverage for Corners

You can optionally enable via coverage analysis for corners. You must use one of the following methods for every corner:

- Include a VIA_COVERAGE_OPTION_FILE command within each corner definition in the corners file. If any corner includes such a file, every corner must include one. If the corners file contains via coverage option files, the StarRC tool ignores global VIA_COVERAGE_OPTION_FILE commands in the command file. Corners with the same nxtgrd file must use the same via coverage option file. However, a single via coverage option file can be used for multiple nxtgrd files.
- Specify via coverage in the nxtgrd files for each corner by using the VIA_COVERAGE option in the source ITF file. If via coverage is specified in the nxtgrd file for any corner, the nxtgrd file for every corner must include it.

The only variations allowed between corners are resistance per via (RPV) variations.

Pattern Density Specification for Corners

You can define different pattern density values outside the design block for different corners by using the DENSITY_OUTSIDE_BLOCK command in the corners file. Simultaneous multicorner extraction supports a maximum of 15 unique combinations of nxtgrd files and DENSITY_OUTSIDE_BLOCK specifications. This feature requires an Ultra+ license.

For calculating the density of a polygon, the StarRC tool considers a 50-micron square window. If the polygon of interest is located near the edge of the block, the final density uses a weighted calculation that takes into account both the actual inside density and the outside density specified with the DENSITY_OUTSIDE_BLOCK command.

If the DENSITY_OUTSIDE_BLOCK command is not set for a corner, the tool applies the value specified by the DENSITY_OUTSIDE_BLOCK command in the StarRC command file. If the StarRC command file does not include a DENSITY_OUTSIDE_BLOCK command, the tool extends the density inside the block to outside the block.

The specified density applies to all layers on which the StarRC tool performs density calculation. The DENSITY_OUTSIDE_BLOCK command has an effect only if the following conditions are met:

- The StarRC command file includes the DENSITY_BASED_THICKNESS: YES command.
- The ITF file contains one or more of the following density-based thickness variation specifications:
  - The THICKNES_VS_DENSITY command
  - The THICKNES_VS_DENSITY_AND_WIDTH command
  - The POLYNOMIAL_BASED_THICKNES_VARIATION command

You can optionally modify the ITF file to include either or both of the USE_SI_DENSITY command, which specifies whether to base the density on drawn or etched dimensions, and the DENSITY_BOX_WEIGHTING_FACTOR command, which changes the dimensions of the analysis window.

RC Scaling for Corners Using Fusion Compiler or IC Compiler II Designs

The following commands can be used in the corners file for NDM format designs created by the Fusion Compiler or IC Compiler II tool:

```
(optional) RES_SCALE: res_value
(optional) CAP_SCALE: cap_value
(optional) CC_SCALE: cc_value
```

The default for the scaling parameters is 1.0, which is equivalent to no scaling. The RES_SCALE factor applies to all resistors, excluding special resistors such as shorting resistors. The CC_SCALE factor applies to all coupling capacitances, excluding coupling capacitances to ground. The CAP_SCALE factor applies to all capacitances including total capacitance, excluding ground capacitances.

Coupling capacitances other than coupling capacitances to ground are subject to both the CC_SCALE and CAP_SCALE factors. Coupling capacitance to ground is adjusted to preserve the total capacitance of the node scaled by the CAP_SCALE value. If the capacitance to ground becomes negative after scaling, it is set to zero.

The scaling parameters are used in the following ways:

- During In-Design extraction in the Fusion Compiler or IC Compiler II tool

  The Fusion Compiler or IC Compiler II tool inserts the scaling commands into the corners file based on settings in the IC Compiler II flow.

- During standalone StarRC extraction

  You can use the scaling commands in a corners file to compare standalone StarRC extraction to Fusion Compiler or IC Compiler II In-Design extraction.

Note:

You can use RC scaling in a standalone StarRC extraction run for comparison with Fusion Compiler or IC Compiler II In-Design extraction runs. However, you should not use RC scaling for final signoff extraction runs.

3-D IC .subckt File for Each Corner

To support corner based 3-D IC .subckt files (specified with the 3D_IC_SUBCKT_FILE command), you can add an attribute in the corners file to support different 3D_IC_SUBCKT_FILE for each corner. You can specify corner specific 3-D IC .subckt files for each corner.

For detailed information, see Specifying Multiple 3-D IC .subckt Files in a Corners File.

The SELECTED_CORNERS Command

The SELECTED_CORNERS command lists the corner names to be extracted. The constraints for the SELECTED_CORNERS command are as follows:

- Each corner name listed in the SELECTED_CORNERS command must match a corner name in the corners file.
- The nxtgrd files associated with the extracted corners must share similar construction.

Use spaces to separate entries in the SELECTED_CORNERS command. Every entry creates one output netlist.

For transistor-level flows, you can write the parasitics from more than one corner into a single netlist, as follows:

- Specify a set of corners to be grouped into a single netlist by using the colon character (:) to separate the corner names. The first corner in a group is considered to be the primary corner for netlist reduction.

  For example, the following command creates one netlist with parasitics for corners C1 and C2 and another netlist with parasitics for corner C4:

  ```
  SELECTED_CORNERS: C1:C2 C4
  ```

- Add an empty corner by using consecutive colons. For example, the following command creates one netlist with parasitics for corners C1 and C4 with an empty field between them:

  ```
  SELECTED_CORNERS: C1::C4
  ```

Corner Constraints

The corners in an SMC flow must obey the following constraints with respect to the ITF commands used to generate the corner nxtgrd files:

- All corners must have the same number and ordering of conducting layers with the same names, WMIN and SMIN values, and T0 values. Conducting layers must have the same covertical configuration. For example, if two conductors are covertical in one corner ITF, they must be covertical in all corners.
- All corners must have the same number and ordering of via layers with the same names, FROM layers, TO layers, and T0 values.
- If any corner conductor or via definition contains a process variation table, the corresponding conductor or via definition in all other corners must contain the same variation table. However, the contents of the tables can be different for each corner.
- All corners must have the same values for the following ITF commands:
  - GATE_TO_CONTACT_SMIN
  - DENSITY_BOX_WEIGHTING_FACTOR
  - HALF_NODE_SCALE_FACTOR
  - GLOBAL_TEMPERATURE

  - USE_SI_DENSITY
  - REFERENCE_DIRECTION

- The following commands are not supported with simultaneous multicorner extraction:
  - DROP_FACTOR (ITF command)
  - RES_UPDATE_FILE (StarRC command)

SMC Output Netlists

By default, netlist file names follow a standard format. You can override the default by specifying the NETLIST_FILE command.

If you do not use the NETLIST_FILE command, the file name format is as follows:

<block>.spf.*<corner>*

The components of the file name are as follows:

- The block name <*block*>, determined as follows:
  - Fusion Compiler, IC Compiler II, or IC Compiler designs: the argument of the StarRC BLOCK command
  - Designs using Hercules or Calibre flows: the argument of the StarRC BLOCK command
  - LEF/DEF designs: the DESIGN keyword in the DEF file specified by the StarRC TOP_DEF_FILE command
- The center part of the file name is fixed as ".spf" regardless of the setting of the NETLIST_FORMAT command.
- The corner name <*corner*> comes from the SELECTED_CORNERS command.

Other StarRC settings affect file creation and naming, as follows:

- Using the NETLIST_COMPRESS_COMMAND command does not change the file name. If you want to use a suffix to indicate file compression (for example, the .gz suffix for the gzip utility), you must use the NETLIST_FILE command to do so.
- By default for gate-level flows, the StarRC tool stores parasitics in the binary parasitic database (GPD) and does not generate an output netlist. To create a netlist during extraction, you must include the NETLIST_FORMAT: SPEF and NETLIST_FILE commands in the StarRC command file. You can also create a netlist from the GPD at a later time.

Feedback

Table 18 shows the file names and file headers that result from SMC extraction.

Table 18 SMC Output Netlist File Names and Headers

NETLIST_FILE	SELECTED_CORNERS	File names	File headers (prefaced with ** for SPF files or // for SPEF files)
<fname>	*<corner1>*	*<fname>.<corner1>*	CORNER_NAME *<corner1>*
not used	*<corner1>*	*<block>*.spf.*<corner1>*	CORNER_NAME *<corner1>*
<fname>	*<corner1> <corner2>*	*<fname>.<corner1>* *<fname>.<corner2>*	CORNER_NAME *<corner1>* CORNER_NAME *<corner2>*
not used	*<corner1> <corner2>*	*<block>*.spf.*<corner1>* *<block>*.spf.*<corner2>*	CORNER_NAME *<corner1>* CORNER_NAME *<corner2>*
<fname>	*<c1>*:*<c2>*	*<fname>.<c1>*:*<c2>*	CORNER_NAME *<c1>*:*<2>*
not used	*<c1>*:*<c2>*	*<block>*.spf.*<c1>*:*<c2>*	CORNER_NAME *<c1>*:*<c2>*

Specifying Multiple 3-D IC .subckt Files in a Corners File

To support corner based 3-D IC .subckt files (specified with the `3D_IC_SUBCKT_FILE` command), you can add an attribute in the corners file to support different `3D_IC_SUBCKT_FILE` for each corner. Make sure that the `CORNERS_FILE` and `SELECTED_CORNERS` commands are also set to perform simultaneous multicorner extraction.

The following example shows the `3D_IC_SUBCKT_FILE` command used to specify files for each corner in the corners file:

```
CORNER_NAME: corner_file_name
TCAD_GRD_FILE: nxtgrd_file1
OPERATING_TEMPERATURE: temperature
3D_IC_SUBCKT_FILE: filename1

CORNER_NAME: corner_file_name
TCAD_GRD_FILE: nxtgrd_file1
OPERATING_TEMPERATURE: temperature
3D_IC_SUBCKT_FILE: filename2

CORNER_NAME: corner_file_name
TCAD_GRD_FILE: nxtgrd_file1
OPERATING_TEMPERATURE: temperature
3D_IC_SUBCKT_FILE: filename3
```

Feedback

Example 10 Output netlist files for corners, generated from different 3-D IC .subckt files

```
*|GROUND_NET 0*|
NET netF 0.0254178PF*|
P (netF B 0 199.5000 0.5000)
Cg4_1 netF 0 1.1211e-14
Cg4_2 netF:2 0 1.13715e-14
Cg4_3 netF:3 0 2.37935e-15
Cg4_4 netF:4 0 4.56e-16
R4_1 netF:3 netF:4 0.12
R4_2 netF:2 netF:4 0.12
R4_3 netF netF:3 0.9
R4_4 netF netF:2 2.42856
C1.sp

*|GROUND_NET 0*|
NET netF 0.0259617PF*|
P (netF B 0 199.5000 0.5000)
Cg4_1 netF 0 1.1211e-14
Cg4_2 netF:2 0 1.13715e-14
Cg4_3 netF:3 0 2.37935e-15
Cg4_4 netF:4 0 9.999e-16
R4_1 netF:3 netF:4 5.0
R4_2 netF:2 netF:4 5.0
R4_3 netF netF:3 0.9
R4_4 netF netF:2 2.42856
C2.spf

*|GROUND_NET 0*|
NET netF 0.0254178PF*|
P (netF B 0 199.5000 0.5000)
Cg4_1 netF 0 1.1211e-14
Cg4_2 netF:2 0 1.13715e-14
Cg4_3 netF:3 0 2.37935e-15
Cg4_4 netF:4 0 4.56e-16
R4_1 netF:3 netF:4 0.130767
R4_2 netF:2 netF:4 0.130767
R4_3 netF netF:3 0.9
R4_4 netF netF:2 3.06989
C3.spf
```

See Also

- 3-D IC .subckt File for Each Corner

Simultaneous Multicorner Flow Examples

Example 1. Three corners with one netlist per corner

The three corners are file *nominal.nxtgrd* analyzed at two temperatures (-25 and 125°C) and file *rcmax.nxtgrd* analyzed at one temperature (25°C).

The StarRC command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: corners.smc
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
```

The corners file contains the following commands:

```
CORNER_NAME: NOM_T1
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: -25

CORNER_NAME: NOM_T2
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: 125

CORNER_NAME: RCMAX_T3
TCAD_GRD_FILE: rcmax.nxtgrd
OPERATING_TEMPERATURE: 25
```

The resulting output files are:

- star.NOM_T1.spf
- star.NOM_T2.spf
- star.RCMAX_T3.spf

Example 2. Three corners with one netlist per corner using the field solver

The following example is the same as Example 1, but using the field solver flow.

The StarRC command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: corners.smc
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
EXTRACTION: FSCOMPARE
```

The resulting output files are:

- *star.fs_comptot.NOM_T1* and *star.fs_compcoup.NOM_T1*
- *star.fs_comptot.NOM_T2* and *star.fs_compcoup.NOM_T2*
- *star.fs_comptot.RCMAX_T3* and *star.fs_compcoup.RCMAX_T3*

Writing Formulas to the Netlist

You can optionally write the RC values from all selected corners into one netlist by writing them as formulas that use the corner names as variables. A simulation tool that reads the

netlist file can set the corner name variables to 1 or 0 to enable or disable the use of the corners.

To enable this option, use the NETLIST_SMC_FORMULA: YES command in the command file and select the corners with the SELECTED_CORNERS command. The allowable netlist formats for this option are SPF, NETNAME, and OA.

Setting the NETLIST_SMC_FORMULA command option to NO (the default) generates netlists according to the format specifications in the SELECTED_CORNERS command.

In the following example, the NETLIST_SMC_FORMULA: YES command is used in a flow in which three corners are defined (NOM_T1, NOM_T2, and RCMAX_T3). The SPF file output contains lines similar to the following example:

```
*|NET A '0.63*NOM_T1+0.65*NOM_T2+0.75*RCMAX_T3'PF
Cg1 A:1 0 3.6e-17*NOM_T1+4.07e-17*NOM_T2+4.09e-17*RCMAX_T3
R162 A:1 A:2 4.8*NOM_T1+4.9*NOM_T2+4.8*RCMAX_T3
```

Writing Resistor Temperature Coefficients to the Netlist

Set the TEMPERATURE_SENSITIVITY command to YES to write the parasitic resistor temperature coefficients TC1 and TC2 to the netlist for use by simulation tools. TC1 and TC2 are obtained from CRT1 and CRT2 in the ITF file. Temperature sensitivity analysis is available only for transistor-level extraction.

The TEMPERATURE_SENSITIVITY: YES command can be used with the simultaneous multicorner flow. The following conditions apply:

- The supported temperature range is -55 C to 150 C.
- The maximum number of selected process corners is 15.
- The OPERATING_TEMPERATURE settings in the corners file are ignored.
- If the selected corners contain redundant nxtgrd files, the tool discards the redundant corners, determines the temperature coefficients for the remaining corners, and issues a warning message.

Temperature coefficients are not reported if they are not set or if they are equal to 0. If the absolute value of TC1 is less than 1e-06, it is set to 1e-06. If the absolute value of TC2 is less than 1e-09, it is set to 1e-09. If the value of TC2 is so small that ignoring it does not significantly affect the resistance error, TC2 might not be written to the netlist.

To report temperature coefficients in SPF, STAR, and NETNAME netlists, the StarRC tool creates a SPICE model card named resStar for parasitic resistors. This model card is

placed inside the .subckt definition for the top-level cell. In the following example, the top cell name is dummy_ABC and the cell has two ports:

```
.subckt dummy_ABC port1 port2
.model resStar R Tref=25
...
.ends
```

The format of the coefficients depends on the netlist type, as follows:

- In the resStar model, temperature coefficients are labeled TC1 and TC2:

  ```
  R1 n1:1 n2:2 resStar R=78 TC1=0.0012556 TC2=-1.95301e-07
  ```

- In a SPEF netlist, temperature coefficients are written using the sensitivity format. The field definitions are written in the *VARIATION_PARAMETERS section of the netlist. The index number of the TC1 and TC2 values might vary in different netlists.

  ```
  *VARIATION_PARAMETERS
  6 CRT1
  7 CRT2 25.0000
  *RES
  1 p1:A p3:Z 2.50093 *SC 4:0.900 5:0.531 6:0.00321 7:-.000021
  ```

The Parasitic Database or GPD

The GPD is a binary database for both gate-level and transistor-level extraction results. The GPD provides these benefits:

- Stores multicorner parasitic data
- Occupies less disk space than other parasitic data formats
- Enables faster data read and write times
- Reduces runtime by eliminating the translation and netlist creation stages
- Can be read directly by other Synopsys tools

The StarRC tool saves extracted parasitic data in a GPD by default for both gate-level and transistor-level flows. If the command file contains unsupported commands, a GPD is not created.

By default, when the tool creates a GPD, an output netlist is not generated. To create a netlist at the time of extraction, specify the `NETLIST_FORMAT` command. You can also generate a netlist directly from a saved GPD at a later time.

Feedback

The following additional GPD usage notes apply to transistor-level flows:

- The IC Validator, Calibre Connectivity Interface, and Virtuoso Integration flows are supported.
- Extraction flows in the Custom Compiler tool are not supported.

For more information, see the following topics:

- Unsupported Commands for GPD Usage
- Creating Other Forms of Output in Addition to a GPD
- The GPD Configuration File
- Additional Notes for Transistor-Level GPD Flows
- The Parasitic Explorer Tool for Querying GPD Contents

Feedback

Unsupported Commands for GPD Usage

Extraction data is saved in the GPD by default.

For all flows, the following commands related to netlist generation are ignored:

- NETLIST_MAX_LINE (causes a warning message)
- NETLIST_POSTPROCESS_COMMAND (causes a warning message)

If any of the commands in Table 19 appear in the command file for a gate-level flow, the StarRC tool issues a warning message and does not create a GPD.

Table 19 Unsupported Commands for Gate-Level Extraction

Unsupported Commands for Gate-Level and General Extraction
3D_IC* commands
CLOCK_NET_INDUCTANCE
CONLY_NETS
COUPLE_NONCRITICAL_NETS* commands
INSTANCE_PORT: CONDUCTIVE MULTIPLE SUFFIXED
NETLIST_COMPRESS_COMMAND other than gzip
NETLIST_POSTPROCESS_COMMAND used with NETLIST_COMPRESS_COMMAND
NETLIST_TAIL_COMMENTS used with VIA_COVERAGE
NETLIST_TAIL_COMMENTS used with SNAP_RESISTOR_WIDTH
METAL_SHEET_OVER_AREA
PIO_FILE
POWER_EXTRACT: RONLY
PROBE_POINTS
SHEET_COUPLE_TO_NET*commands
SKIP_CELLS_COUPLE_TO_NET*commands
ZONE_COUPLE_TO_NET*commands

If any of the commands in Table 19 or Table 20 appear in the command file for a transistor-level flow, the StarRC tool issues a warning message and does not create a GPD.

Table 20 Unsupported Commands for Transistor-Level Extraction

Unsupported Commands for Transistor-Level Extraction
CONVERT_DIODE_TO_PARASITIC_CAP
ITF_FILE
MERGE_INSTANCE_PORTS
MERGE_PARALLEL_DEVICES
NETLIST_HIER_PROBE_NODES
NETLIST_PARASITIC_RESISTOR_MODEL
NETLIST_PRINT_CC_TWICE
NETLIST_SIM_OPTIONS
NETLIST_USE_M_FACTOR
TARGET_PWRA: YES
USER_DEFINED_DIFFUSION_RES
XREF: COMPLETE

Creating Other Forms of Output in Addition to a GPD

You can create other forms of output directly from a GPD either during the extraction run or at a later time.

Creating a SPEF Netlist

You can create a Standard Parasitic Exchange Format (SPEF) netlist directly from the parasitic database, either during the extraction run or at a later time.

- To create a SPEF netlist during the extraction run, include the `NETLIST_FORMAT: SPEF` and `NETLIST_FILE` commands in the command file.
- To create a SPEF netlist from a saved GPD directory, use the following command, where *gpd_dir* is the GPD directory and *spef_file* is the name to use for the output SPEF file:

  ```
  % StarXtract -convert_gpd_to_spef gpd_dir spef_file
  ```

  If the StarRC command file used for the original extraction run contained commands for distributed processing, distributed processing is also used when creating a SPEF netlist from the GPD.

Note:

SPEF files created from a GPD database do not have routing_conf or driver_reduction sections to represent delay values for pi networks.

Feedback

Creating an SPF Netlist

You can create a Standard Parasitic Format (SPF) netlist directly from the parasitic database, either during the extraction run or at a later time.

- To create an SPF netlist during the extraction run, include the `NETLIST_FORMAT: SPF` and `NETLIST_FILE` commands in the command file.
- To create an SPF netlist from a saved GPD directory, use the following command, where *gpd_dir* is the GPD directory and *spf_file* is the name to use for the output SPF file:

  ```
  % StarXtract -convert_gpd_to_spf gpd_dir spf_file
  ```

  If the StarRC command file used for the original extraction run contained commands for distributed processing, distributed processing is also used when creating an SPF netlist from the GPD.

Creating an OpenAccess View

You can create an OpenAccess (OA) view directly from the parasitic database, either during the extraction run or at a later time. This option is valid only for transistor-level flows.

- To create an OA view during the extraction run, include the `NETLIST_FORMAT: OA` and `NETLIST_FILE` commands in the command file.
- To create an OA view from a saved GPD directory, use the following command, where *gpd_dir* is the GPD directory and *oa_file* is the name of the OA configuration file, which includes the OA library path and other necessary settings:

  ```
  % StarXtract -convert_gpd_to_oa gpd_dir oa_file
  ```

Feedback

The GPD Configuration File

After an extraction is complete, you can modify selected aspects of the stored data by using the GPD configuration file.

Follow this procedure:

1. Execute a StarRC extraction run that creates a GPD.
2. Generate the configuration file by executing the following command, where *gpd_dir* is the name of the GPD directory:

   ```
   % StarXtract -dump_gpd_config gpd_dir
   ```

3. Edit the configuration file.
4. Apply the edited configuration file to the GPD database:

   ```
   % StarXtract -set_gpd_config gpd_dir config_file
   ```

5. Use the modified database to generate a new SPEF netlist:

   ```
   % StarXtract -convert_gpd_to_spef gpd_dir spef_file
   ```

6. (Optional) Reset the GPD to its original configuration:

   ```
   % StarXtract -reset_gpd gpd_dir
   ```

GPD Configuration File Commands

If the new configuration conflicts with the existing configuration, the tool issues an error message. For example, you cannot specify a coupling threshold that is smaller than the threshold used in the original extraction.

The following commands are allowed in a GPD configuration file:

- `COUPLING_ABS_THRESHOLD`
- `COUPLING_REL_THRESHOLD`
- `COUPLING_THRESHOLD_OPERATION`
- `GPD_DP_STRING`

  This command is valid only in a GPD configuration file. If you are creating a SPEF netlist from a GPD, you can use this command in the configuration file to specify distributed processing conditions for netlist creation that are different from the distributed processing conditions used in the extraction run. Distributed processing during extraction is controlled by the `STARRC_DP_STRING` command.

- NETLIST_COMPRESS

  This command is valid only in a GPD configuration file.

- NETLIST_CONNECT_SECTION
- NETLIST_DEVICE_LOCATION_ORIENTATION
- NETLIST_DELIMITER
- NETLIST_GROUND_NODE_NAME
- NETLIST_INCREMENTAL
- NETLIST_NAME_MAP
- NETLIST_NODE_SECTION
- NETLIST_PASSIVE_PARAMS
- NETLIST_PRECISION
- NETLIST_SELECT_NETS
- NETLIST_SUBCKT
- NETLIST_TYPE
- SELECTED_CORNERS

  The corners in the configuration file must be a subset of the corners extracted in the original run.

Additional Notes for Transistor-Level GPD Flows

The following sections describe differences between transistor-level and gate-level GPD flows.

Output With SPICE Subcircuit Files

The output of the GPD flow is different from the output of a standard (xout) flow when the SPICE_SUBCIRCUIT_FILE command is used and the SPICE file contains fewer ports than the layout. For example, consider a layout that contains ports a, b, c, and d, but the SPICE subcircuit file contains only ports a, b, and c. An SPF output file from a standard flow is as follows:

```
.SUBCKT cell a b c
*|P a
*|P b
*|P c
*|P d
```

However, if you create an SPF file from a GPD saved from the same extraction, the output is as follows:

```
.SUBCKT cell a b c
*|P a
*|P b
*|P c
*|IS node_name
```

Effects of the REDUCTION: HIGH Command on Output Netlists

When you set the `REDUCTION` command to `HIGH` in a standard flow (without a GPD), the StarRC tool performs reduction in two steps, the first during extraction and the second during netlisting. In a GPD flow, the `REDUCTION: HIGH` command affects output netlists as follows:

- SPEF netlists

  If you create a SPEF netlist either during the extraction run or later from the saved GPD, the SPEF netlist size is larger than a SPEF netlist generated during a standard extraction run (without a GPD). The tool cannot perform the second reduction step due to the way node IDs are stored internally.

- SPF netlists

  If you create an SPF netlist either during the extraction run or later from the saved GPD, the netlist size of the SPF netlist is the same as an SPF netlist created in a standard extraction run (without a GPD). However, you can only create an SPF netlist from the saved GPD. You cannot create a SPEF netlist from the saved GPD.

Specific Path Selection

For a GPD created from a transistor-level flow, you can select specific paths and subsequently generate an output file or OA view that includes only those paths. This provides faster turnaround time than working with the entire design when you are using a downstream simulator to debug specific issues.

Use the following command syntax:

```
StarXtract -gpd_select_path <gpd_dir> <path_file_name>
```

The path file is a text file in which every line is a path selection written as follows:

```
select_path: <start_point> <end_point>
```

The start and endpoints can be ports or instances. Use the following format for a port:

```
P:<port_name>
```

Use the following format for an instance:

`I:<instance_name>`

Wildcards are supported for port and instance names. Acceptable paths are from port to port, instance to instance, and port to instance. All paths between the startpoint and endpoint are retrieved.

To create the GPD before selecting specific paths, run a full-chip extraction with the `NETS:*` and `POWER_EXTRACT: YES` commands to ensure that the network in the GPD is complete.

The Parasitic Explorer Tool for Querying GPD Contents

The Parasitic Explorer tool allows you examine the contents of a GPD. You can use interactive commands, a Tcl language script, or a graphical user interface to perform operations such as querying parasitics on specific nets, generating custom reports, and investigating shorts.

For more information, see the *Parasitic Explorer User Guide*.

Creating a GPD for Parasitic Explorer Tool Use

The StarRC user guide lists commands that are not supported for creating a GPD. If you use any unsupported commands during extraction, a GPD is not created and you cannot use the Parasitic Explorer tool.

When you create a GPD intended for later use with the Parasitic Explorer tool, you must set the `PARASITIC_EXPLORER_ENABLE_ANALYSIS` command to `YES` during the extraction to ensure that the GPD contains the necessary information.

Some StarRC commands are acceptable for creating a GPD, but are not compatible with the Parasitic Explorer tool. Observe the following guidelines:

- The `SHORT_PINS: NO` command is not supported.
- The `REDUCTION` command affects the values and locations of the reported parasitics. Set the command to `NO` or `LAYER_NO_EXTRA_LOOPS` for optimum correspondence of the parasitics to the input database.

Transistor-level GPDs intended for later use with the Parasitic Explorer tool must adhere to the following requirements:

- The `REMOVE_FLOATING_NETS` command must be set to `YES`.
- The `XREF` command must be set to `YES`.

- The `TRANSLATE_RETAIN_BULK_LAYERS` command must be set to `CONLY` to avoid creating multiple substrate nodes.
- The `XREF_LAYOUT_NET_PREFIX` command cannot specify a prefix that contains special characters. The default prefix of ln_ is recommended.

Saving Data for Displaying Layout Information Around Shorts

The Parasitic Explorer tool provides a user interface for displaying design objects in the vicinity of shorts discovered during extraction. To ensure that information about specific shorts is available for the Parasitic Explorer tool, you can create a file that contains the additional layout information only for specified nets or regions.

Use one of the following methods during the extraction:

- Use the `-write_short_regions` option with the `StarXtract` command. For example:

  ```
  %StarXtract -write_short_regions -nets_file file_name cmd_file
  ```

  The nets file contains a list of net names separated by spaces or line breaks.

- Specify a region of interest by using the `-window` option. The arguments *llx*, *lly*, *urx*, and *ury* are the lower-left x-coordinate, lower-left y-coordinate, upper-right x-coordinate, and upper-right y-coordinate. For example:

  ```
  %StarXtract -write_short_regions -window llx lly urx ury cmd_file
  ```

The `-nets_file` and `-window` options are mutually exclusive.

Saving Parasitic Resistor Attributes

The StarRC command file controls whether some properties of parasitic resistors are stored in the GPD during extraction. The following commands affect parasitic resistor attributes:

- The `NETLIST_TAIL_COMMENTS: YES` command stores the following attributes:
  - is_via
  - is_via_array
  - length
  - width
- The `EXTRA_GEOMETRY_INFO: RES` command stores the following attributes:
  - x_coordinate_max
  - x_coordinate_min

    - y_coordinate_max
    - y_coordinate_min
- Running simultaneous multicorner extraction by using the `SIMULTANEOUS_MULTI_CORNER: YES` command stores the following attributes:
    - resistance_max
    - resistance_min
    - resistance_multicorner
- Running single-corner extraction stores the following attribute:
    - resistance

Feedback

Directories and Files

The following topics describe directories and reports that are created by or used by the StarRC tool during extraction:

- The Star Directory
- The Summary File
- Coordinate Scaling in the Netlist and Summary Reports
- Shorts Reports
- Opens Reports
- SMIN Violations Reports
- Via Violations Reports
- Metal Fill Reports
- Coupling Capacitance Reports
- Hierarchical Coupling Reports
- The Power Nets Report
- DEF File Override Reports
- FSCOMPARE Flow Output Files

The Star Directory

The `STAR_DIRECTORY` command specifies a directory that the StarRC tool creates for reports and intermediate files. The command defaults to the string "star." As a result, StarRC documentation often uses the term "star directory" as a generic term. However, you can use any string that follows valid directory naming conventions.

The `STAR_DIRECTORY` command accepts both absolute and relative paths. However, you can specify a relative path only if the directory is below the working directory (the working directory is the directory from which you run the `StarXtract` command). For example:

```
% star_dir/working_dir/other_dir     (incorrect)
% working_dir/other_dir/star_dir     (correct)
```

Feedback

The Summary File

The SUMMARY_FILE command specifies the name of the summary file, which is generated by the tool and contains information, warning, and error messages that occur during the run. The file also reports the elapsed time, CPU time, and peak memory usage.

By default, the summary file is located in the run directory and has the name *block_name*.star_sum, where *block_name* is the block specified by the BLOCK command.

You can use the SUMMARY_FILE command to change the name and location of the summary file. This command accepts a path relative to the run directory. Absolute paths are not permitted.

The following command creates a summary file my_summary.log in the results subdirectory:

```
SUMMARY_FILE: ./results/my_summary.log
```

Coordinate Scaling in the Netlist and Summary Reports

Unscaled (original layout) coordinates are useful for investigating opens, shorts, and SMIN or via violations.

If the StarRC command file contains a MAGNIFICATION_FACTOR command, the coordinates in the *opens.sum*, *shorts_all.sum*, *smin.sum,* and *vias.sum* files are scaled by default. To report unscaled coordinates instead, set the NETLIST_UNSCALED_COORDINATES command to YES. Table 21 shows the effect of ITF and StarRC commands on coordinate scaling for these reports.

Table 21 Coordinate Scaling For Netlist and Summary Reports

HALF_NODE_SCALE_FACTOR	MAGNIFICATION_FACTOR	NETLIST_UNSCALED_COORDINATES	Coordinates
value	value	ignored	n/a (error)
not set	value	YES	unscaled
not set	value	NO (default)	scaled
value	not set	not set (StarRC sets to YES automatically)	unscaled
value	not set	YES	unscaled
value	not set	NO	scaled

Feedback

Shorts Reports

Features in the same layer that abut or overlap are potential shorts. The StarRC tool reports potential shorts in a file named *shorts_all.sum* in the star directory. The tool always reports a default set of shorts.

The *shorts_all.sum* file contains information about blockages and skip cells by default. You can disable the additional information by setting the CELLS_IN_SHORTREPORT command to NO. You can expand the types of shorts included in the *shorts_all.sum* file by setting the ENHANCED_SHORT_REPORTING command to YES or COMPLETE.

To limit the number of unique shorts reported in the *shorts_all.sum* file, use the FILL_SHORTS_LIMIT command for fill shorts and the SHORTS_LIMITcommand for other types of shorts. If the limit is exceeded, the StarRC tool writes a warning message in the file and does not report the additional violations. The default for both limits is 1000.

A typical short between a signal net and a blockage or skip cell is reported in the following format:

```
Short between net <name1> and <name2> of instance <top/cell> of cell
 <macro> Layer = <layer> BBox=(<coords>)
```

The following lines are examples of shorts in the *shorts_all.sum* file:

```
Short between net vss and net UNSIG_295 of instance top1/cell1 of cell
 macro1 Layer = M6  Bbox=(444.204, 427.342), (444.204,427.387)
Short between net vss and unselectable net Layer = M6 Bbox=(...)
Short between net vss and blockage Layer = M6 Bbox=(...)
```

For information about allowing the escape (\) character in the net names of the shorts summary file, see VIOLATION_REPORT_SPEF_ESCAPING.

If polygons of two different nets cross each other, one of the polygons is cut into three pieces: one consisting of the overlap area and two that abut the overlap area. As a result, the StarRC tool reports three shorts: one from the overlap area and two from the abutting polygon segments. The abutting shorts are reported with zero width and nonzero length.

In addition, the tool creates a file named *shorts_by_cell.sum*. This file contains all the instances reported in the *shorts_all.sum* file. Each line corresponds to one instantiated cell and reports the cell name, the number of shorts within the cell, and the names of the shorted nets within the cell. The nets are sorted by the number of shorts, in decreasing order. Only the top ten nets (by number of shorts) are reported.

The format is as follows:

```
Instantiated_cell Nb_shorts Top_ten_nets <cell_name> <no_of_shorts>
 <net0> <net1> ... <net10>
```

Feedback

For example:

```
Instantiated_cell Nb_shorts Top_ten_nets cell0 6 cell0/n1 cell0/n2
 cell0/n12 cell0/n14 cell0/n20 cell0/n21
```

Table 22 describes the terminology associated with conductor polygons.

Table 22 Conductor Definitions

Conductor type	Definition
Extracted signal net	Metal assigned to an extracted signal net
Power net	Metal assigned to a power net when the StarRC command `POWER_EXTRACT` is set to `NO`
Unselectable signal net	Metal assigned to a ground, dangling, or floating net that is unselectable depending on the StarRC options
Non-extracted signal net	Metal assigned to a signal net (not a power or ground net) that is not specified for extraction
Floating fill	Fill metal set to floating potential
Grounded fill	Fill metal set to ground potential
Blockage	Placeholder for a block to be added to the layout later
Skip cell	A cell treated as gray-box material (other than the ports) for extraction

Table 23 lists the shorts that are reported in the *shorts_all.sum* file for the settings of the `ENHANCED_SHORT_REPORTING` command.

Table 23 Shorts Reported For Settings of the ENHANCED_SHORT_REPORTING Command

First polygon	Second polygon	NO	YES	COMPLETE
Extracted signal net	Extracted signal net	yes	yes	yes
Extracted signal net	Unselectable signal net	no	yes	yes
Extracted signal net	Non-extracted signal net	no	yes	yes
Extracted signal net	Floating fill	yes	yes	yes
Extracted signal net	Grounded fill	yes	yes	yes
Extracted signal net	Blockage polygon	no	yes	yes

Feedback

Table 23 Shorts Reported For Settings of the ENHANCED_SHORT_REPORTING Command (Continued)

First polygon	Second polygon	NO	YES	COMPLETE
Extracted signal net	Skip cell ports of extracted nets	yes	yes	yes
Extracted signal net	Skip cell internal nets	no	no	yes
Port of extracted net	Extracted signal net	yes	yes	yes
Port of extracted net	Non-extracted signal net	no	no	no

See Also

- Coordinate Scaling in the Netlist and Summary Reports

Opens Reports

By default, the StarRC tool identifies segments of an open net and connects them by inserting one or more shorting resistors. Open nets are reported in a file named *opens.sum* located in the STAR directory. Shorting resistors are reported in the netlist along with parasitic resistors.

The tool detects opens by recognizing layout shapes that have the same net name or net number but are not physically connected. Each segment of an open net is called a resistively connected group (RCG). Inserting resistors makes it possible for most timing analysis tools to calculate delays, even though one or more nets are not actually connected.

The intended connectivity of the open net is not known. The StarRC tool uses geometric proximity and layer compatibility to select the nodes of an RCG between which to insert a shorting resistor.

By default, the tool connects all open nets. You can control this behavior by using the `NETLIST_CONNECT_OPENS` command to specify which nets the tool should connect.

Shorting resistors have a resistance of 0.01 ohms and a width of 100 units to make them easy to recognize in the parasitic netlist. Settings of the `REDUCTION` and `EXTRACTION` commands might cause the coordinates of the shorting resistors to vary.

Feedback

The contents of the opens.sum report and the output netlist are as follows:

- Shorting resistors
  - The output netlist includes the shorting resistors. For example,

    ```
    R2_35 vdd2:2 vdd2 0.01
    ```

  - The *opens.sum* file contains entries similar to the following (coordinates in microns):

    ```
    OPEN net: NETA (2 RCGs):
       shorting resistor: 14997 (resistance=0.01,width=100) is inserted
     between nodes ABC_MUX72:1 (located at (0.1800,-0.3663, layer=M1)
     and ABC_MUX72:2 (located at (0.1800,0.3213), layer=M2(floating))
    ```

- No shorting resistors
  - The output netlist has incomplete connectivity for those nets.
  - The *opens.sum* file contains a notice similar to the following:

    ```
    WARNING: NETLIST_CONNECT_OPENS is not set for the following open
    net(s) thus they will not appear in the "opens.sum" file. In order
    to see information on the shorting resistors in the "opens.sum"
    file, you need to run -cleanN using NETLIST_CONNECT_OPENS:

    vdd2
    net_xyz
    ...
    ```

For information about allowing the escape (\) character in the net names of the opens summary file, see VIOLATION_REPORT_SPEF_ESCAPING.

See Also

- Coordinate Scaling in the Netlist and Summary Reports

Feedback

SMIN Violations Reports

By default, the StarRC tool does not report SMIN violations. An SMIN violation between two neighboring polygons in a conductor layer occurs if all of the following conditions are met:

- The drawn distance between the polygons is less than the SMIN value.
- The polygons belong to different nets.
- The polygons are not floating fill polygons or via clones.
- No more than one of the polygons is a grounded fill polygon.
- The distance over which the SMIN spacing is in violation is at least 10 times the WMIN value.

To create a report of SMIN violations, set the REPORT_SMIN_VIOLATION command to YES. The tool creates a file named *smin.sum* in the star directory. In addition, the tool writes warning messages in the summary file to indicate that SMIN violations occur and to direct you to see the *smin.sum* file for more information.

By default, the REPORT_SMIN_VIOLATION command is set to NO, in which case the *smin.sum* file is not created and the summary file does not contain warnings about SMIN violations.

You can limit the number of unique SMIN violations reported in the *smin.sum* file by setting the SMIN_LIMIT command. If the limit is exceeded, the StarRC tool writes a warning message in the file and does not report the additional violations. The default is 1000.

For each SMIN violation, the *smin.sum* report lists the two nets involved, the bounding box, and the layer name. In the report, a net exhibits violations with respect to all of the nets whose names are indented in the subsequent lines. Each pair of nets might have multiple locations where an SMIN violation occurs, which is indicated in the report by multiple bounding boxes.

As a result, every violation appears two times in the report, one time under each of the net names. The following lines are examples from an SMIN violation report:

```
SIGNALNET
  LEFT0
   at BBox=(911.11,-0.133),(880.455,-0.047) on M2,M2
   at BBox=(199.999,-0.133),(177.702,-0.047) on M2,M2
   ...
  RIGHT0
   at BBox=(977.234,0.047),(955.555,0.133) on M2,M2
   at BBox=(933.332,0.047),(910.185,0.133) on M2,M2
   ...

RIGHT0
```

```
SIGNALNET
 at BBox=(977.234,0.047),(955.555,0.133) on M2,M2
 at BBox=(933.332,0.047),(910.185,0.133) on M2,M2
 ...
```

See Also

- Coordinate Scaling in the Netlist and Summary Reports

Via Violations Reports

If the design contains vias with only one connection or with no connections, the StarRC tool creates a file named *vias.sum* in the star directory.

You can limit the number of unique via violations reported in the *vias.sum* file by setting the `VIA_SUM_LIMIT` command. If the limit is exceeded, the StarRC tool writes a warning message in the file and does not report the additional violations. The default is 1000.

Examples of entries in the *vias.sum* file are as follows:

```
Via with one or no connection at BBox=(1.2340,1.5165),(1.2520,1.5525)
of net n10 on layer diffCont
Via with one or no connection at BBox=(1.2340,1.3185),(1.2520,1.35455)
of net n10 on layer diffCont
```

Metal Fill Reports

You can obtain statistics and debugging information about the metal fill features in your design by setting the `REPORT_METAL_FILL_STATISTICS` command to `YES`. Enhanced reporting is available for metal fill that comes from GDSII or OASIS files.

Note:

Generating metal fill statistics incurs a runtime penalty.

Table 24 summarizes which reports are generated for combinations of commands and metal fill sources. Table 25 describes the file contents for advanced metal fill reporting.

Table 24 Metal Fill Summary Report Generation

REPORT_METAL_FILL_STATISTICS	Metal fill source	Number of fill cells	Summary files	Contents of mf_statistics.sum
NO	Any	Any	None	Not present
YES	GDSII or OASIS	< 1000	mf_statistics.sum	Advanced reporting: complete

Feedback

Table 24 Metal Fill Summary Report Generation (Continued)

REPORT_METAL_FILL_STATISTICS	Metal fill source	Number of fill cells	Summary files	Contents of mf_statistics.sum
YES	GDSII or OASIS	>= 1000	mf_statistics.sum mf_cells.sum mf_cell_polygon_count.sum	Advanced reporting: summary with pointers to other files
YES	Other	Any	mf_statistics.sum	Standard reporting

Table 25 File Contents For Advanced Metal Fill Reporting

File name	< 1000 fill cells	>= 1000 fill cells
mf_statistics.sum	Number of fill polygons per layer Names of cells read in Error and warning messages Number of polygons per layer for each cell	Number of fill polygons per layer Types of errors and warnings encountered Pointers to other summary files
mf_cells.sum	Not generated	Names of cells read in Error and warning messages
mf_cell_polygon_count.sum	Not generated	Number of polygons for each cell

The number of information, warning, and error messages reported in the summary files is limited by the settings of the `MESSAGE_SUPPRESSION` and `MESSAGE_SUPPRESSION_LIMIT` commands.

Examples of mf_statistics.sum File For Fill Cells < 1000

When the number of metal fill cells is less than 1000, the *mf_statistics.sum* file contains the complete report. For example:

```
<database layer>  <no. of fill polygons>
M1                      53156
M2                     109199
M3                     155626
C4                     111502
...

Total 63 metal fill cells are imported. (SX-2165)
Cell definition for 'A" is read in from 'xyz.gds'. (SX-3033)
Duplicate cell definition for 'A' is found and ignored in 'top.gds'.
(SX-2021)
Cell definition for 'B' is read in from 'xyz.gds'. (SX-3033)
```

```
...
WARNING: Undefined GDS/OASIS cell 'C' (SX-2550)

Cell A:
     M1                  2760

Cell B:
     M1                 20004
     M2                 12152
     M3                 32190
     ...
```

Example of mf_statistics.sum File For Fill Cells >= 1000

When the number of metal fill cells is 1000 or more, the results are divided between the *mf_statistics.sum*, *mf_cells.sum*, and *mf_polygon_count.sum* files. For example:

```
<database layer>  <no. of fill polygons>
M1                   53156
M2                  109199
M3                  155626
C4                  111502
...

Duplicate cell definition is found and ignored in the GDS/OASIS files.
Some cells are not defined in the GDS/OASIS files.
Total 1220 metal fill cells are imported. (SX-2165)
Total 1220 cells are read in from GDS/OASIS files.

For details, refer to the summary/mf_cells.sum

For polygon count of each cell StarRC read, refer to
 summary/mf_cell_polygon_count.sum
```

Example of mf_cells.sum File

The following is an example of the *mf_cells.sum* file:

```
Duplicate cell definition for 'STN' is found and ignored in
'xyz.gds'. (SX-2021)
Duplicate cell definition for 'ABC' is found and ignored in
'xyz.gds'. (SX-2021)
...
GDS/OASIS cell 'B2' is not defined (SX-2550)
GDS/OASIS cell 'B3' is not defined (SX-2550)
...
Cell 'Cell_A1' is read-in from 'xyz.gds' (SX-3033)
Cell 'B2' is read-in from 'xyz.gds' (SX-3033)
Cell 'MUX_13' is read-in from 'jkl.gds' (SX-3033)
...
```

Feedback

Example of mf_polygon_count.sum File

The following is an example of the *mf_polygon_count.sum* file:

```
Cell         PolygonCount          ReferenceCount         TotalCount

cell_A               1000                      20               1020
cell_B               1000                      20               1020
cell_C               1000                      20               1020
...
```

Coupling Capacitance Reports

You can generate optional coupling capacitance reports as follows:

- Specify a file name in the `COUPLING_REPORT_FILE` command to generate a report that lists the coupling capacitance by net. The report is sorted by the ratio of coupling capacitance to total capacitance for the net. The report uses the following format:

  *Cc/Ct *100 Cc victim_net aggressor_net*

  The number of entries in the report is limited by the `COUPLING_REPORT_NUMBER` command.

  The total net capacitance used for the coupling percentage calculation is the net capacitance that is used for smart decoupling. It does not include loading pin capacitors and intranet coupling capacitors (same net coupling).

  ```
  * 1000 worst couplings in descending order
  * ratio(%) coupling victim aggressor
  30.00 3e-15 net1 net2
  20.00 2e-15 net3 net2
  10.00 3e-15 net2 net1
  ```

- Set the `NONCRITICAL_COUPLING_REPORT_FILE` command to a file name to generate a report that lists all coupling capacitances from non-extracted signal nets to critical nodes, including the following:

  - A comment at the top of the file refers to the corresponding SPEF file name, *prefix* command, and *suffix* command.
  - The report lists all coupling capacitances from noncritical nets to critical nodes, in reverse order from the SPEF file output. The following example shows the SPEF file and report file output for the same object:

    ```
    SPEF:
    14 A B/SYNOPSYS_INCONTEXT_b 1.0e-15
    ```

```
Report file:
14 B/SYNOPSYS_INCONTEXT_b A 1.0e-15
```

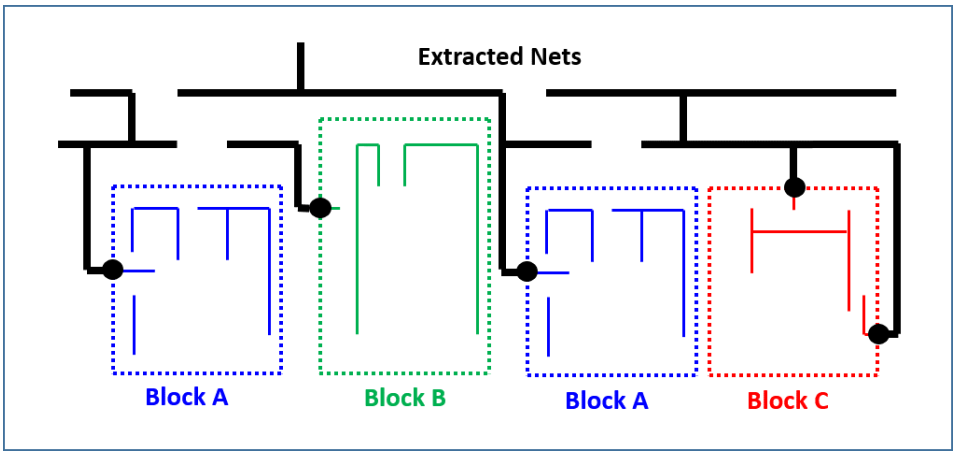
- The command works with `NETLIST_NAME_MAP:YES | NO` for net name mapping of noncritical net names.

Hierarchical Coupling Reports

Set the `HIERARCHICAL_COUPLING_REPORT_FILE` command to a file name to generate a report that lists the coupling capacitances between extracted signal nets and skipped cells.

Figure 32 illustrates a design that contains four skip cells. By default, features in skip cells are treated as ground during extraction. However, the coupling capacitance between extracted signal nets and skip cells can be significant and might affect timing analysis. Analyzing hierarchical coupling capacitance provides insight into this occurrence.

Figure 32 Relationship Between Extracted Nets and Skip Cell Nets



The hierarchical coupling capacitance includes the capacitance to all signal nets inside the skip cell instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

The hierarchical coupling capacitance report presents coupling capacitances in decreasing order of the relative percentage of coupling capacitance to total capacitance for any extracted signal net. The report contains the number of entries specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command.

Feedback

Hierarchical coupling capacitance is reported if the following conditions are all true:

- The HIERARCHICAL_COUPLING_REPORT_FILE command is present, which enables the feature and specifies a file name for the report.
- The absolute capacitance is larger than the value specified by the HIERARCHICAL_COUPLING_ABS_THRESHOLD command.
- The relative capacitance, calculated as a percentage of the extracted signal net capacitance, is larger than the value specified by the HIERARCHICAL_COUPLING_REL_THRESHOLD command.
- The number of reported capacitances is smaller than the value specified by the HIERARCHICAL_COUPLING_REPORT_NUMBER command. Hierarchical capacitances are reported in decreasing order of their relative capacitance values.

In a simultaneous multicorner flow, the coupling report is generated only for the primary corner.

The following example shows the contents of a report file:

```
* 637 worst coupling capacitances listed in descending order:
* relative      coupling         extracted name    instance name
  weight (%)    capacitance      net name          name
   23.2          1.386e-15       I237/B77          I20 (adder1)
   21.8          2.733e-15       T336/W22/I90      I31 (INV2)
   16.4          5.33e-14        PACKAGE_3         I20 (adder1)
   15.1          3.773e-15       GB_32772          I17 (mux2d4)
       ...
```

In this example, critical signal net I237/B77 has 23.2 percent of its total capacitance coupling to signal nets in skip cell instance I20. The magnitude of the coupling capacitance is 1.386e-15 Farads.

The Power Nets Report

To save memory and runtime, the StarRC tool does not extract power nets by default. However, you can use the POWER_EXTRACT command to specify that some or all power nets should be extracted.

The StarRC tool creates a file named *power_net_names* that contains the list of all nets identified to be power nets. Net names are written in fully expanded form, even if wildcards are used in the POWER_NETS command.

For transistor-level flows, names come from the schematic domain when the XREF command is set to YES or COMPLETE, and from the layout domain when the XREF command is set to NO.

The list of power nets is not affected by the setting of the POWER_EXTRACT command. Power nets are identified by labels in the design database, by the POWER_NETS command, or both.

The power nets file is an alphabetical list of nets. As shown in the following example, the file begins with the line begin_power_nets and ends with the line end_power_nets:

```
begin_power_nets
A
GND
VSS
VSSA
end_power_nets
```

DEF File Override Reports

By default, the StarRC tool uses LEF attributes from the default LEF file for DEF macros. The DEF_ATTRIBUTE_FROM_LEF command obtains one or more LEF attributes from the specified LEF file and overwrites the defaults of those attributes for the specified DEF macro.

You can specify the DEF_ATTRIBUTE_FROM_LEF command multiple times in a command file. However, a specific DEF macro can obtain properties from only one LEF file.

The following four attributes can be obtained from the LEF file: WIDTH, VIA, NDR, and WRONGDIRECTION.

When you use the DEF_ATTRIBUTE_FROM_LEF command, a directory named DEFoverride is automatically created under the STAR directory. For every macro specified for an override, the tool creates a summary file listing all of the attributes that are overwritten from the specified LEF file.

To generate a report that includes the override information from all instances of the DEF_ATTRIBUTE_FROM_LEF command, set the DEF_OVERRIDE_REPORT_FILE command to a file name. The report file is located in the run directory and includes the macro DEF file name, the override LEF file name, and a list of the property types and property names that are replaced.

An example of the override report is as follows:

```
MACRO Cell Name : MACROCELL1
LEF FILE : tech1.lef

VIA VIA12
VIA VIA23
VIA VIA56
NDR NDR_TEST

==========================================
```

```
MACRO Cell Name : periph_2
LEF FILE : .../tech/foundry_mxb2.lef

VIA VIA7045
NDR NDR_3p
```

Feedback

FSCOMPARE Flow Output Files

The `EXTRACTION: FSCOMPARE` command initiates a flow that reports differences between pattern-based StarRC extraction and field solver extraction.

The `FSCOMPARE` flow generates a report with the extension *.fs_comptot* for general capacitances and a report with the extension *.fs_compcoup* for coupling capacitances. The nets included in these reports are filtered by the values of the `FSCOMPARE_THRESHOLD` command for general capacitances and the `FSCOMPARE_COUPLING_THRESHOLD` command for coupling capacitances.

In both the *.fs_comptot* and *.fs_compcoup* reports, shorted nets are listed separately from unshorted nets. The first section (the correlation report) excludes shorted nets and includes overall statistics. The second section lists the shorted nets, showing the per-net correlation and the short type, but no overall statistics.

Output files generated in the `FSCOMPARE` flow include a detail section for SMIN violations nets similar to the detail section for shorted nets. In addition, the files provide a message if the violation limit has been reached and the name of the summary file that contains more information.

See Also

- Output Files for the FSCOMPARE Flow

Feedback

Output Netlists

The StarRC tool can create several types of parasitics netlists for use as input to other tools. Each of them contains a slightly different structure and format. This section provides a simple example of each netlist type.

The available netlist types are as follows:

- Standard Parasitic Exchange Format (SPEF)

  This is the format most commonly used for timing analysis.

- Standard Parasitic Format (SPF)

  This format contains device-level information for input to simulation tools.

- NETNAME format (transistor-level extraction only)

  This is a SPICE-style output for input to simulation tools. Internal node names are included, which provides more information about the nets to which the parasitic elements are attached.

- OpenAccess format (transistor-level extraction only)

  OpenAccess (OA) is an open standard data exchange format for circuit design.

- STAR format (transistor-level extraction only)

  STAR format is a SPICE-style output for input to simulation tools.

See Also

- SPEF Netlist Example
- SPF Netlist Example
- NETNAME Netlist Example
- NETLIST_IDEAL_SPICE_FILE Output Example
- STAR Netlist Example

SPEF Netlist Example

```
*SPEF "IEEE 1481-1999"
*DESIGN "ALU"
*DATE "Wed April 17 19:50:14 2006"
*VENDOR "Synopsys"
*PROGRAM "StarRC"
*VERSION "K-2015.06"
```

```
*DESIGN_FLOW "PIN_CAP NONE" "NAME_SCOPE LOCAL"
*DIVIDER /
*DELIMITER :
*BUS_DELIMITER []
*T_UNIT 1 NS
*C_UNIT 1 FF
*R_UNIT 1 OHM
*L_UNIT 1 HENRY

*NAME_MAP
*5 net1
*10 insta/net2
*14 U1
*16 insta/U1
*17 insta/U2

*PORTS
*5 I *C 3.6 0

*D_NET *5 1.02e+00

*CONN
*P *5 I *C 3.6 0
*I *14:I I *C 6.76 8.94 *L 4 *D inv

*CAP
1 *5 0.60481
2 *14:I 0.413795

*RES
1 *5 *14:I 29.8492

*END

*D_NET *10 4.58e-01

*CONN
*I *16:ZN O *C 9.12 21.84
*I *17:I I *C 6.34 20.94 *L 4 *D inv

*CAP
1 *16:ZN 0.271701
2 *17:I 0.186

*RES
1 *16:ZN *17:I 16.8989

*END
```

Feedback

SPF Netlist Example

```
*
*|DSPF 1.0
*|DESIGN top
*|DATE "Wed April 17 13:34:43 2000"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "2003.2.0.0"
*|DIVIDER /
*|DELIMITER :
*

.SUBCKT top net1

*|GROUND_NET 0

*|NET net1 9.60e-04PF
*|P (net1 I 0 3.6 0)
*|I (U1:I U1 I I 4e-15 6.76 8.94)
R1 net1 U1:I 29.8492
Cg1 net1 0 5.8737e-16
Cg2 U1:I 0 3.72441e-16

*|NET insta/net2 5.10e-04PF
*|I (insta/U1:ZN insta/U1 ZN O 0 9.12 21.84)
*|I (insta/U2:I insta/U2 I I 4e-15 6.34 20.94)
R2 insta/U1:ZN insta/U2:I 16.8989
Cg3 insta/U1:ZN 0 2.96927e-16
Cg4 insta/U2:I 0 2.13328e-16
*
* Instance Section
*
Xinsta/U2 insta/U2:I in2 VDD VSS inv
Xinsta/U1 in1 insta/U1:ZN VDD VSS inv
XU1 U1:I net2 VDD VSS inv

.ENDS
```

NETNAME Netlist Example

```
*
*|DSPF 1.3
*|DESIGN toprt
*|DATE "Thu April 10 13:00:11 2001"
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "2003.2.0.0"
*|DIVIDER /
*|DELIMITER :
**FORMAT STAR
```

```
*

*|GROUND_NET 0

*|NET min_msb_led[0] 1.06e-02PF
*|P (min_msb_led[0] O 0 0 425.8)
*|I (min_msb_led[0]:F485 min_msb/conv_blk1/U4 X O 0 37 394)
Rmin_msb_led[0] min_msb_led[0] min_msb_led[0]:F1558 0.001
Cg1 min_msb_led[0]:F485 0 3.17002e-15
Cg2 min_msb_led[0]:F1558 0 7.44086e-15
R1 min_msb_led[0]:F485 min_msb_led[0]:F1558 12.105

*|NET min_lsb/cnt_blk1/n200 1.00e-02PF
*|I (min_lsb/cnt_blk1/n200:F191 min_lsb/cnt_blk1/U51 X O 0
63.9 49.8)
*|I (min_lsb/cnt_blk1/n200:F141 min_lsb/cnt_blk1/U53 C I 5e-
14 117 53)
Cg3 min_lsb/cnt_blk1/n200:F191 0 5.58775e-15
Cg4 min_lsb/cnt_blk1/n200:F141 0 4.42815e-15
R2 min_lsb/cnt_blk1/n200:F191 min_lsb/cnt_blk1/n200:F141
10.1364
*
* Instance Section
*
Xmin_lsb/cnt_blk1/U39 min_lsb/n100:F162 min_lsb/cnt_blk1/
n185:F164 VDD VSS INV2
Xmin_msb/cnt_blk1/U44 VDD min_msb/cnt_blk1/n216:F521 VDD
VSS INV2
Xmin_lsb/cnt_blk1/U45 min_lsb/n100:F235 min_lsb/cnt_blk1/
n195:F239 min_lsb/cnt_blk1/n190:F237 VDD VSS AND2
Xsec_msb/conv_blk1/U33 sec_msb/conv_blk1/n68:F1471
sec_msb_led[2]:F1475 sec_msb/conv_blk1/n67:F1473 VDD VSS
OR2
Xsec_msb/conv_blk1/U29 sec_msb/conv_blk1/n52:F1476
sec_msb/conv_blk1/n65:F1480 sec_msb/bcd[2]:F1478 VDD VSS
OR2
```

NETLIST_IDEAL_SPICE_FILE Output Example

```
* SPICE Netlist
* VENDOR "Synopsys, Inc."
* PROGRAM "StarRC"
* DATE "Thu April 16 16:26:00 2002"

**FORMAT SPICE

.SUBCKT AND2 B A OUT
XS1I1 B A S1N3 NAND2
XS1I2 OUT S1N3 INVA
.ENDS AND2
```

```
.SUBCKT AND3 C B A OUT
XS1I1 C B A S1N9 NAND3
XS1I2 OUT S1N9 INVA
.ENDS AND3

.SUBCKT CS_ADD1 SUM COUT C B A
XS1I2 B A S1N29 AND2
XS1I3 C S1N9 S1N31 AND2
XS1I4 S1N43 S1N35 S1N39 AND2
XS1I5 S1N29 S1N31 S1N35 NOR2
XS1I6 S1N41 S1N39 S1N37 NOR2
XS1I7 C B A S1N41 AND3
XS1I8 C B A S1N43 OR3
XS1I16 B A S1N9 OR2
XS1I33 COUT S1N35 INVA
XS1I34 SUM S1N37 INVA
.ENDS CS_ADD1

.SUBCKT INVA OUT IN
MS1I1 OUT IN GND GND N ad=39p as=39p l=1u pd=32u ps=32u w=13u
MS1I2 VDD IN OUT VDD P ad=61.5p as=61.5p l=1u pd=47u ps=47u
w=20.5u
.ENDS INVA

*.SUBCKT N D G S VBB
*.ENDS N

.SUBCKT NAND2 B A QN
MS1I1 S1N20 A GND GND N ad=13p as=39p l=1u pd=15u ps=32u w=13u
MS1I3 VDD B QN VDD P ad=61.5p as=46.125p l=1u pd=47u ps=25u
w=20.5u
MS1I4 VDD A QN VDD P ad=61.5p as=46.125p l=1u pd=47u ps=25u
w=20.5u
MS1I25 QN B S1N20 GND N ad=39p as=13p l=1u pd=32u ps=15u w=13u
.ENDS NAND2

.SUBCKT NAND3 C B A QN
MS1I1 S1N11 A GND GND N ad=19.5p as=39p l=1u pd=16u ps=32u
w=13u
MS1I4 VDD A QN VDD P ad=61.5p as=41p l=1u pd=47u ps=24.5u
w=20.5u
MS1I28 VDD B QN VDD P ad=35.875p as=41p l=1u pd=24u ps=24.5u
w=20.5u
MS1I29 VDD C QN VDD P ad=35.875p as=61.5p l=1u pd=24u ps=47u
w=20.5u
MS1I30 S1N32 B S1N11 GND N ad=19.5p as=19.5p l=1u pd=16u
ps=16u w=13u
MS1I31 QN C S1N32 GND N ad=39p as=19.5p l=1u pd=32u ps=16u
w=13u
.ENDS NAND3

.SUBCKT NOR2 B A QN
MS1I1 QN A GND GND N ad=29.25p as=39p l=1u pd=17.5u ps=32u
```

```
w=13u
MS1I2 QN B GND GND N ad=29.25p as=39p l=1u pd=17.5u ps=32u
w=13u
MS1I3 S1N5 B QN VDD P ad=20.5p as=112.75p l=1u pd=22.5u
ps=52u w=20.5u
MS1I4 VDD A S1N5 VDD P ad=61.5p as=20.5p l=1u pd=47u ps=22.5u
w=20.5u
.ENDS NOR2

.SUBCKT NOR3 C B A QN
MS1I1 QN A GND GND N ad=26p as=39p l=1u pd=17u ps=32u w=13u
MS1I2 QN B GND GND N ad=26p as=22.75p l=1u pd=17u ps=16.5u
w=13u
MS1I3 S1N5 B S1N25 VDD P ad=30.75p as=30.75p l=1u pd=23.5u
ps=23.5u w=20.5u
MS1I4 VDD A S1N5 VDD P ad=61.5p as=30.75p l=1u pd=47u ps=23.5u
w=20.5u
MS1I41 S1N25 C QN VDD P ad=30.75p as=82p l=1u pd=23.5u ps=49u
w=20.5u
MS1I42 QN C GND GND N ad=39p as=22.75p l=1u pd=32u ps=16.5u
w=13u
.ENDS NOR3

.SUBCKT OR2 B A OUT
XS1I1 B A S1N3 NOR2
XS1I2 OUT S1N3 INVA
.ENDS OR2

.SUBCKT OR3 C B A OUT
XS1I1 C B A S1N3 NOR3
XS1I2 OUT S1N3 INVA
.ENDS OR3

*.SUBCKT P D G S VBB
*.ENDS P

*.SUBCKT ADD4 S1N9 S1N7 S1N5 SUM3 SUM2 SUM1 SUM0 COUT CIN
B3 B2 B1 B0 A3 A2 A1 A0
XS1I1 SUM0 S1N5 CIN B0 A0 CS_ADD1
XS1I2 SUM1 S1N7 S1N5 B1 A1 CS_ADD1

XS1I3 SUM2 S1N9 S1N7 B2 A2 CS_ADD1
XS1I4 SUM3 COUT S1N9 B3 A3 CS_ADD1
*.ENDS ADD4
```

STAR Netlist Example

```
*
*|DSPF 1.3
*|DESIGN top
*|DATE "Wed April 17 13:38:52 2000"
```

Feedback

```
*|VENDOR "Synopsys"
*|PROGRAM "StarRC"
*|VERSION "2003.2.0.0"
*|DIVIDER /
*|DELIMITER :
*

.SUBCKT top net1

*|GROUND_NET 0
*|NET net1 9.60e-04PF
*|P (net1 I 0 3.6 0)
*|I (F5 U1 I I 4e-15 6.76 8.94)
*|S (F6 3.6 0)
Rnet1 net1 F6 0.001
R1 F6 F5 29.8492
Cg1 F6 0 5.8737e-16
Cg2 F5 0 3.72441e-16

*|NET insta/net2 5.10e-04PF
*|I (F2 insta/U1 ZN O 0 9.12 21.84)
*|I (F4 insta/U2 I I 4e-15 6.34 20.94)
R2 F2 F4 16.8989
Cg3 F2 0 2.96927e-16
Cg4 F4 0 2.13328e-16
*
* Instance Section
*
Xinsta/U2 F4 in2 VDD VSS inv
Xinsta/U1 in1 F2 VDD VSS inv
XU1 F5 net2 VDD VSS inv

.ENDS
```

Netlists For HSIM Reliability Analysis

Because you can set multiple commands in a StarRC command file when extracting a design for HSIM reliability analysis, often designers specify more design parameters than are needed. This leads to high memory use and large netlists. To remedy this, you can use the `TARGET_PWRA:YES` command to generate a netlist relevant to the reliability analysis flow. You need only specify the power nets in the command file.

Creating a Simplified Command File

To create the simplified command file, specify the commands as shown in the following example:

```
BLOCK:
MILKYWAY_DATABASE:
MILKYWAY_EXTRACT_VIEW:
TARGET_PWRA:YES
```

```
POWER_NETS: list_of_power_nets
TCAD_GRD_FILE:
NETLIST_INSTANCE_SECTION: YES | ALL
MAPPING_FILE:
XREF: YES
SKIP_CELLS:
```

Using the NETLIST_INSTANCE_SECTION command forces the tool to include devices connected to an unextracted power net in the netlist.

The TARGET_PWRA command automatically includes the commands needed for power reliability analysis and overrides any commands of the same type that appear elsewhere in the command file. The only exception is the POWER_REDUCTION command. The recommended option is the LAYER_NO_EXTRA_LOOPS option; the TARGET_PWRA command sets this option if no other instance of the POWER_REDUCTION command appears in the command file. If you set the POWER_REDUCTION command to YES in the command file, the TARGET_PWRA command overrides it with the LAYER_NO_EXTRA_LOOPS option. However, if you set the POWER_REDUCTION command to the more conservative NO or LAYER options, those settings are not changed.

The TARGET_PWRA: YES command causes the StarRC tool to generate two netlists. One netlist contains unreduced resistors for power nets. The other netlist contains reduced RC-coupled devices for signal nets. The signal netlist is useful for both reliability analysis and signal timing analysis.

If you use the TARGET_PWRA command, you must also use the REDUCTION: LAYER and KEEP_VIA_NODES: NO commands.

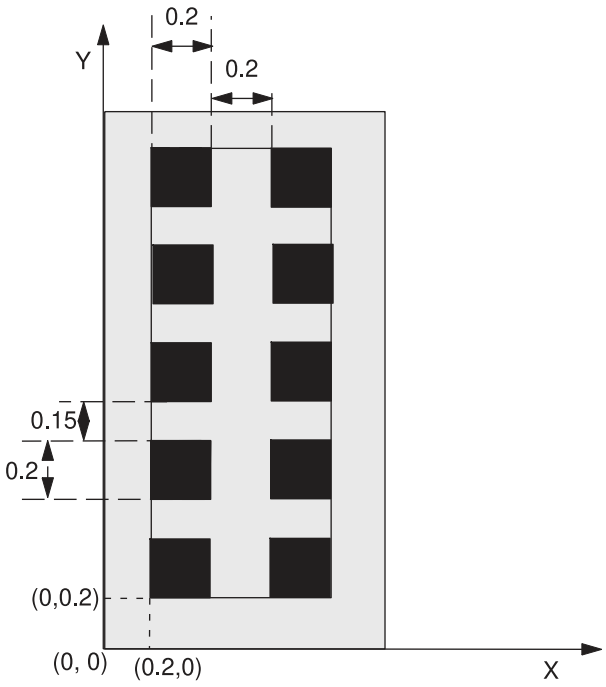
SPF Geometry Visualization in HSIM

To see the effects of merged vias in an SPF file, you must include the NETLIST_TAIL_COMMENTS: YES, EXTRA_GEOMETRY_INFO: NODE, and KEEP_VIA_NODES: YES commands in the command file.

Figure 33 shows a 5x2 via array. The individual via size is 0.2 microns by 0.2 microns, the vertical via-to-via spacing is 0.15 microns, and the horizontal via-to-via spacing is 0.2 microns. If you specify MAX_VIA_ARRAY_SPACING=0.3 in the mapping file for the via layer, this via array extracted as one via resistor.

Feedback

Figure 33 Via Array



If the `NETLIST_TAIL_COMMENTS` command is set to `YES`, an output SPF file contains the following information:

```
R45 29 32 0.2 $a=0.4 $lvl=5 $n=5x2 $p=4.4
```

Nodes 29 and 32 reside on the upper and lower layers connected by the via resistor. The via resistor value is calculated based on the total area of the vias, which is reported as the `$a` parameter. The `$p` parameter is the perimeter of the bounding box of the via array and is calculated as follows:

perimeter = (0.2 x 2 + 0.2) x 2 + (0.2 x 5 + 0.15 x 4) * 2 = 4.4

Extraction For Electromigration Analysis

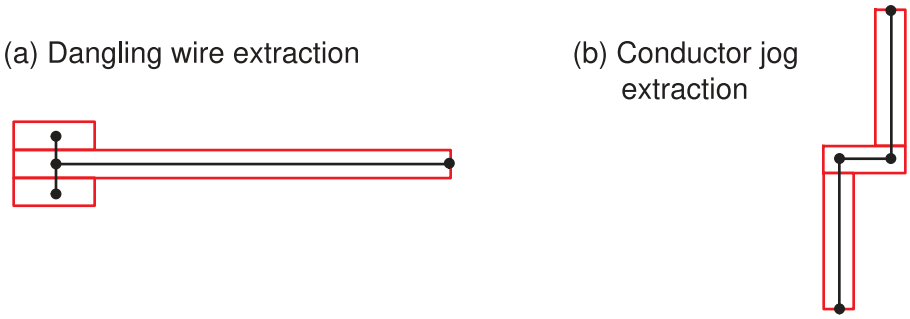
Electromigration refers to the detrimental physical effects of high current densities on integrated circuit features. As process geometries shrink, the effects of electromigration become more important.

Simulation tools that analyze electromigration risk use information from StarRC transistor-level extraction runs. The location, resistance, and width of parasitic resistors are of primary importance.

Feedback

Certain StarRC extraction methods are included for the purpose of providing accurate input for subsequent electromigration analysis. For example, the tool extracts the resistance for dangling wires. In Figure 34(a), a horizontal resistor is extracted for the dangling wire in addition to the two vertical resistors. The tool also accurately represents the current direction and resistance of conductors that have small jogs, as shown in Figure 34(b).

Figure 34 Extraction Modifications for Electromigration Analysis



To generate a parasitics netlist to use with an electromigration analysis tool, observe the following recommendations:

- Use the following settings in the StarRC command file:
  - `REDUCTION: NO`
  - `EXTRA_GEOMETRY_INFO: NODE RES`
  - `KEEP_VIA_NODES: YES`
  - `SHORT_PINS: NO`
  - `NETLIST_TAIL_COMMENTS: YES`
  - `NETLIST_NODE_SECTION: YES`
  - `NETLIST_CONNECT_SECTION: YES`
- To preserve a network node or subnode for individual vias, do not use via merging in the `via_layers` section of the layer map file. In other words, do not set either the `MAX_VIA_ARRAY_SPACING` or the `MAX_VIA_ARRAY_LENGTH` command in the `via_layers` section.

Feedback

The Default Netlist Format

If you do not provide a mapping file, the netlist contains the default set of parameters. The default netlist format for layer resistors is as follows.

```
1 *318 *1:4 0.33 // $l=0.8000 $w=1.6000 $lvl=2 $llx=584.4000 $lly=0.8000
    $urx=586.0000 $ury=0.0000 $dir=1
```

Table 26 describes the default parameters. For via resistors, the area is reported instead of the length and width.

Table 26 Default Netlist Parameters

Label	Description
$l	Length (microns; layer resistors only)
$w	Width (microns; layer resistors only)
$a	Area (square microns; via resistors only)
$lvl	Level
$llx	Lower-left x-coordinate (microns)
$lly	Lower-left y-coordinate (microns)
$urx	Upper-right x-coordinate (microns)
$ury	Upper-right y-coordinate (microns)
$dir	Direction of current flow (0 for horizontal, 1 for vertical, 2 for non-Manhattan)

The Electromigration Parameter Mapping File

You can change the parameter labels and specify which parameters to include in the netlist by specifying a mapping file with the `EM_PARAM_MAPPING_FILE` command.

Note:

The `EM_PARAM_MAPPING_FILE` command and the specified mapping file are for flows that use third-party electromigration (EM) and reliability analysis (RA) tools. Do not use this command with flows that use the Synopsys CustomSim tool.

The EM parameter mapping file uses the following syntax, where the two values are separated by one or more spaces. Denote a comment line with an initial pound sign (#).

```
# Comment line
PARAMETER_NAME     user_label
```

The first entry is one of the fixed parameter names. The second entry is a user-specified label to be written to the parasitic netlist for that parameter. Table 27 lists the available electromigration parameters.

The order of the parameters in the EM parameter mapping file does not matter. In the parasitic netlist, the order of the parameters that are reported for each device is fixed. However, some parameters might not be present, depending on the contents of the EM parameter mapping file.

Table 27 Parameters Available in Electromigration Parameter Mapping File

StarRC parameter name	Description
RES_WIDTH	Resistor width in microns
RES_LENGTH	Resistor length in microns
RES_LAYER	Resistor layer number, which corresponds to the layer number in the LAYER_MAP section of the netlist
RES_NONPHYSICAL_LAYER	Name for a new layer which contains all shorting resistors and which appears in the layer map section of the netlist
RES_VIA_AREA	Via area for a resistor on a via
RES_VIA_NUM	For a via array, the number of vias; for a trench contact virtual via, the reciprocal of the number of via segments
RES_MASK_ID	Mask ID of the layer if the resistor is on a contact layer with multiple masks
RES_CENTER_X	The x-coordinate of the center of the resistor
RES_CENTER_Y	The y-coordinate of the center of the resistor
RES_BBOX_LLX	The x-coordinate of the lower-left corner of the resistor bounding box
RES_BBOX_LLY	The y-coordinate of the lower-left corner of the resistor bounding box
RES_BBOX_URX	The x-coordinate of the upper-right corner of the resistor bounding box
RES_BBOX_URY	The y-coordinate of the upper-right corner of the resistor bounding box

Consider the following EM parameter mapping file:

```
RES_LENGTH      L
RES_WIDTH       W
RES_LAYER       lvl
RES_CENTER_X    X
RES_CENTER_Y    Y
```

This mapping file produces lines in a parasitic netlist similar to the following.

```
R3_1 VDD:1 VDD:2 1.05597 $m1 $L=5.3000 $W=1.0200 $lvl=13 $X=42.8550
$Y=811.200
```

All resistors display the ITF layer name (in the format $<*layer_name*>). The LAYER_MAP section of the netlist contains the ITF layer names.

5

ECO Extraction

ECO extraction is the technique of performing extraction only on parts of a design that are different from a reference design. This capability allows efficient evaluation of engineering change orders, especially when coupled with timing analysis and place and route tools that work together.

For more information, see the following topics:

- ECO Extraction Overview
- The StarRC Incremental ECO Flow

Feedback

ECO Extraction Overview

For timing closure and crosstalk analysis, iterative design and analysis runs are frequently necessary to identify and fix problems. The design changes are known as engineering change orders (ECOs). ECO changes might include localized gate placement and sizing changes, net rerouting, and net additions or deletions. After a design modification, parasitic extraction and timing analysis must be repeated to verify that the issues are resolved. Repeating the full-chip extraction and analysis cycle can be very time-consuming.

The StarRC incremental ECO flow reduces turnaround time by performing extraction only on the ECO-affected nets. In this flow, the Fusion Compiler, IC Compiler II, StarRC, and PrimeTime tools work together to share information about ECO design changes.

This flow supports Fusion Compiler, IC Compiler II, Milkyway, and LEF/DEF designs.

Note:

If you enable ECO extraction by setting the `ECO_MODE` command to `YES` or `RESET`, you must also use the `GPD` command to specify a GPD directory name.

ECO extraction is compatible with the following StarRC features:

- Distributed processing
- Simultaneous multicorner analysis

The following usage notes apply to ECO extraction:

- SPF netlists are not supported.
- Field solver extraction is not supported.
- Only RC extraction is supported. The `EXTRACTION` command must be set to `RC`.
- Power net extraction is not supported. The `POWER_EXTRACT` command must be set to `NO`.
- The `COUPLE_TO_GROUND` command must be set to `NO`.

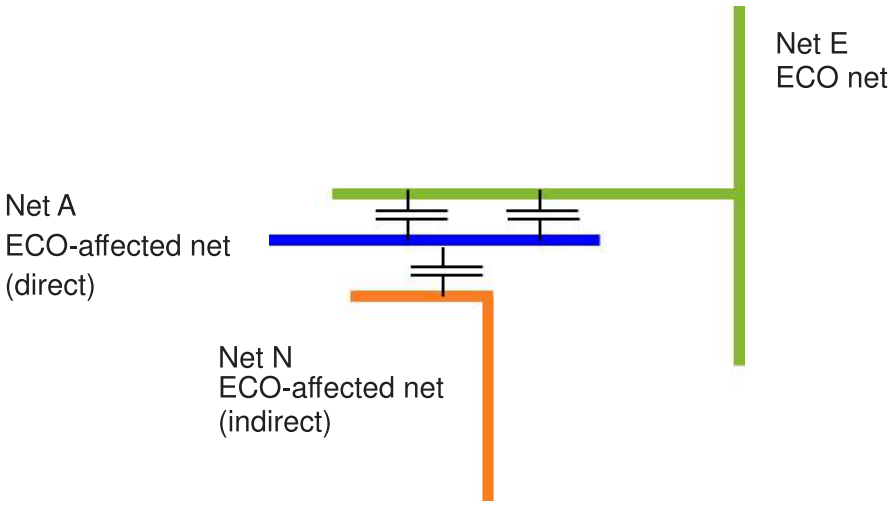
Feedback

Identification of Nets Affected by an ECO

Figure 35 shows the types of nets that might be affected by a design change. Net E, the ECO net, is a net that is modified as part of a timing violation fix. Net A is coupled to net E; this type of net is a directly ECO-affected net. Net N is not coupled directly to net E, but is coupled to net A. This type of net is an indirectly ECO-affected net.

The StarRC tool includes net E and net A, but not net N, in the list of affected nets. The coupling capacitance between net A and net N is considered to be a dangling capacitance under net A. The PrimeTime tool can adjust the coupling and total capacitance of net N accordingly.

Figure 35 ECO-Affected Nets



If a cell is swapped with a footprint-compatible cell without changing the instance name and connected wires, the StarRC tool does not re-extract the connected wires or replace them in the netlist. The PrimeTime tool identifies the correct cell name for timing analysis based on the instance name in the ECO Verilog file.

Feedback

The StarRC Incremental ECO Flow

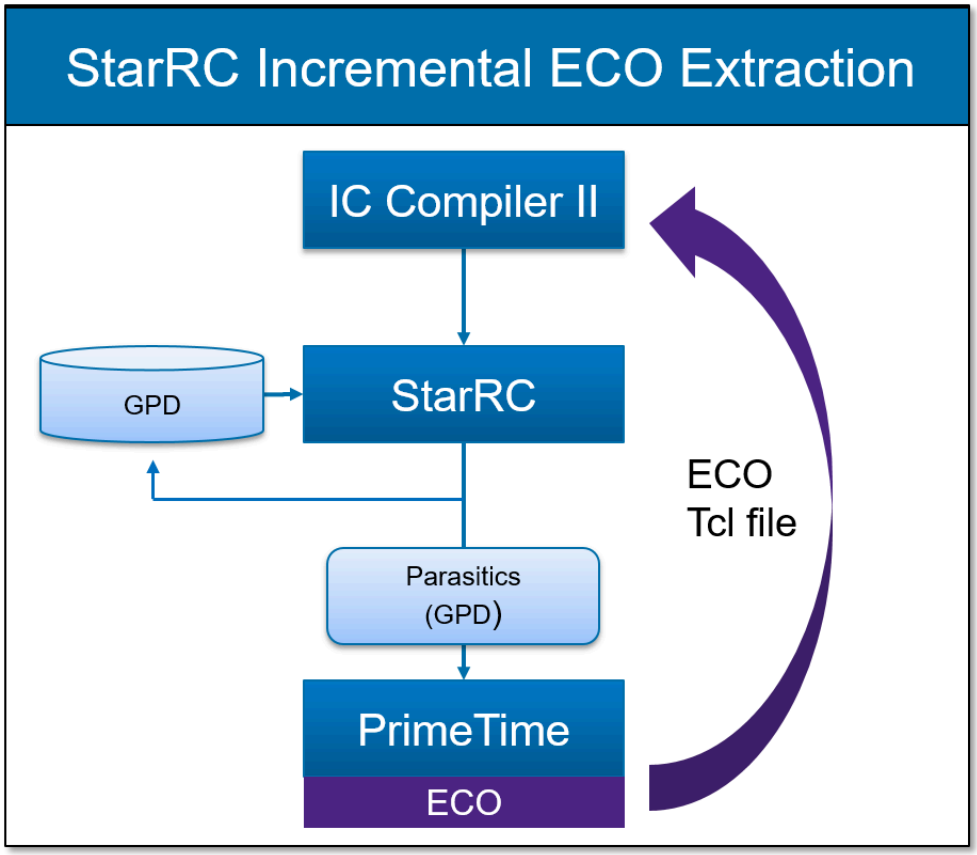
In this flow, the Fusion Compiler, IC Compiler II, StarRC, and PrimeTime tools work together to share information about ECO design changes. The combined flow offers the largest overall turnaround time benefit.

For more information about procedures and commands in the Fusion Compiler and IC Compiler II tools, see the *Fusion Compiler Implementation User Guide* and *IC Compiler II Implementation User Guide* respectively. For more information about procedures and commands in the PrimeTime tool, see the *PrimeTime User Guide*.

Figure 36 illustrates the incremental extraction ECO flow.

- The StarRC tool performs either full-chip extraction or incremental extraction and saves the results in the GPD and, optionally, in a SPEF file.
- The PrimeTime tool performs either full-chip or incremental timing analysis, using the GPD as input.

Figure 36 ECO Extraction and Signoff Flow



Feedback

This procedure is an overview of the ECO flow.

1. Create a design in the Fusion Compiler or IC Compiler II tool. Milkyway and LEF/DEF designs are also supported.
2. Include the following commands in the StarRC command file. The commands and file names are examples for an IC Compiler II design; modify them as needed. In these examples, the specified GPD directory name includes the characters .gpd as a reminder of the directory contents.

   For an IC Compiler II or a Fusion Compiler design using the GPD:

   ```
   NDM_DATABASE: design1.ndm
   BLOCK: top_block_rev0
   GPD: eco_full_chip.gpd
   ECO_MODE: YES
   STAR_DIRECTORY: star
   ```

3. Execute a baseline extraction run with the StarRC tool.

   The first run is a full-chip extraction. The parasitic data is saved in the GPD in the directory specified by the `GPD` command.

4. Execute a baseline full update timing run with the PrimeTime tool.

   Use the PrimeTime `read_parasitics` command with the GPD directory name or SPEF file name. To use the GPD from a StarRC simultaneous multicorner extraction run, you must specify a corner with the PrimeTime `set_parasitic_corner_name` command.

   To use the GPD:

   ```
   pt_shell> read_parasitics -format gpd -keep_capacitive_coupling   \
                             eco_full_chip.gpd
   pt_shell> set parasitic_corner_name <corner_name>
   pt_shell> update_timing -full
   ```

5. In the PrimeTime tool, specify ECO design changes to address timing violations and save the changes to a Tcl script file.
6. Implement the ECO changes and save the design. You can optionally use a new name for the modified design.
7. Copy the GPD directory to a new directory to prepare for the next ECO iteration.
8. Modify the StarRC command file as follows:
   - Edit the `NDM_DATABASE` and `BLOCK` commands, if needed to point to the modified design. If you are using a Milkyway or LEF/DEF design, modify the associated StarRC commands.
   - Edit the `GPD` command to point to the new GPD directory.

Feedback

- (Optional) Edit the `STAR_DIRECTORY` command to point to the new star directory.
- (Optional) Edit the `SUMMARY_FILE` command to change the file name. Otherwise, the previous summary file is overwritten.

9. Execute a StarRC ECO extraction run on the modified design.

   The extraction might be either a full-chip extraction or an ECO extraction, depending on the amount of design changes and the runtime benefit determined by the StarRC tool. However, the GPD always contains the full-chip parasitic data.

10. Modify the PrimeTime script as follows:
    - Edit the `read_parasitics` command to point to the latest GPD directory.
    - To perform incremental timing analysis instead of full-chip timing analysis, remove the `-full` option from the `update_timing` command.
11. Execute the timing run and check for timing violations.
12. Repeat steps 5 to 11 until timing closure is achieved.
13. For chip signoff, perform a final full-chip extraction and timing analysis.

6

The StarRC Field Solver

The StarRC field solver is a 3-D capacitance extraction tool that solves electrostatic equations using statistical random-walk methods. The field solver is used to verify the accuracy of StarRC results or to provide enhanced extraction accuracy for selected nets.

For more information, see the following topics:

Feedback

Overview of Field Solver Extraction

The field solver extracts capacitance by solving the three-dimensional Laplace equations. By contrast, standard pattern-based extraction compares layout structures to a library of structures for which capacitances have already been determined. The field solver provides greater accuracy at the expense of runtime.

You can control the accuracy and runtime of the field solver. In addition, the tool can run on a single CPU, multiple threads within the same CPU, or in distributed processing mode.

Capacitance Types

The field solver reports the following capacitances at a node:

- Total capacitance – The capacitance from the net to all other objects in the simulation.
- Coupling capacitance (xcap) – The capacitance between two nets that are not part of the same net group.

Each capacitance type is reported in a separate section of the field solver capacitance report.

Boundary Conditions

Boundary conditions are applied at the six edges of the simulation domain. By default, the field solver uses a Dirichlet boundary condition of 0 volts. This is equivalent to placing a ground plane along each face of the simulation cube. These ground planes are connected to the field solver global ground net.

In general, the total node capacitance obtained with the Dirichlet boundary condition used in the field solver is higher than the total capacitance obtained with Neumann boundary conditions used in the RC2 and RC3 modes of the Raphael tool. As the boundary moves further from the simulated electrodes, the effect becomes smaller.

You can select Neumann boundary conditions in the field solver by adding the `-neuman_x` and `-neuman_y` options to the `FSCOMPARE_OPTIONS` command in the StarRC command file. For example,

```
FSCOMPARE_OPTIONS: -neuman_x -neuman_y
```

You also can specify periodic boundary conditions on the x- or y-direction with the `-periodic_x` and `-periodic_y` options to the `FSCOMPARE_OPTIONS` command. Periodic boundary conditions cause the random walks to wrap around to the opposite side of the device. For example, a walk that exits the device at the positive x-side reenters at the

negative x-side of the device. Periodic boundary conditions are useful for devices with repeated cells such as memory circuits. For example,

```
FSCOMPARE_OPTIONS: -periodic_x -periodic_y
```

Conductor Types

Structures in the field solver are composed of conductors and dielectrics. Dielectrics are typically derived from the technology file. Conductors are composed of metal boxes or planar boxes.

The field solver uses the following conductor types:

- Nets
- Net Groups
- Ground Nets
- Fill Nets

Conductors are composed of collections of metal boxes. All the metal boxes in a conductor are connected and at the same potential (voltage). The following sections describe the types of conductors and their reporting rules.

Nets

A net conductor is the most common general type of conductor. Random walks are started only from nets and net groups. The field solver reports capacitances between nets and from nets to all other types. Nets are created by placing metal boxes or planar boxes between a net statement and an end statement. Net statements are specified in the encrypted design file to describe and identify the nets.

Net Groups

A net group is a collection of nets that are electrically connected and therefore have the same potential (voltage). The field solver extracts the capacitance between nets in different net groups but not between nets in the same net group. Net groups are specified in the design file.

Ground Nets

If Dirichlet boundary conditions are used as the default, the edges of the bounding box are electrical ground planes that form a single net called ground. In addition, in the design file, some nets might be identified as ground. The field solver reports the capacitance from other nets to ground, but because analysis walks are not started from ground, the total capacitance of ground is not reported.

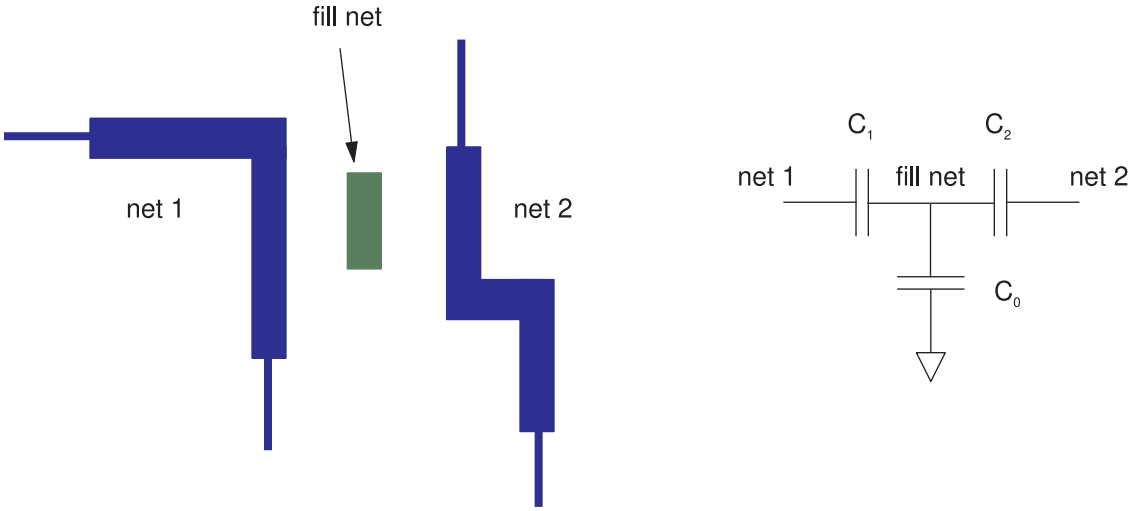
Feedback

Fill Nets

Fill nets are used to model fill metal polygons. Metal fill consists of boxes inserted into a metal layer to improve the planarity of the metal layer. A fill net is electrically floating—that is, not connected to any circuit elements in the netlist.

Even though fill nets are not electrically connected, they can introduce capacitive coupling effects between other nets. For example, in Figure 37, net 1 and net 2 are signal nets. A fill polygon is placed between net 1 and net 2.

Figure 37 Example of Fill Nets



The charge on the fill net is calculated as shown in Equation 1.

Equation 1 Calculation of Charge on a Fill Net

$$Q_1 = \frac{V_1 C_1 C_0 + (V_1 - V_2) C_1 C_2}{C_0 + C_1 + C_2}$$

The effective capacitance at net 1 is calculated as shown in Equation 2.

Equation 2 Calculation of Effective Capacitance Between Fill Nets

$$C_{eff} = \frac{C_1 C_0 + C_1 C_2}{C_0 + C_1 + C_2}$$

The coupling capacitance between net 1 and net 2 is calculated as shown in Equation 3.

Equation 3 Calculation of Coupling Capacitance

$$C_c = \frac{C_1 C_2}{C_0 + C_1 + C_2}$$

Feedback

Advantages of Using the Field Solver Flow

The Field Solver flow efficiently provides accurate extraction information in a design. You can get the following results with the use of the field solver flow:

- Extract all nets or a list of nets in a design within the specified percentage value.
- Extract library of structures or develop a library structure, during the initial stage of cell characterization to evaluate the performance of a process for design technology co-optimization (DTCO) applications.
- Apply boundary conditions on memory cells that are arrayed in a design or on memory cells that must be characterized to consider the effect of memory cells during the characterization flow.
- Looks at larger distance to extract capacitance between nets that are far away from each other.

Running the Field Solver

The StarRC tool includes two methods of using field solver extraction:

- The `FSCOMPARE` flow

  Reports the differences between the pattern-based extraction results of the StarRC tool and the random-walk extraction results of the field solver

- The `FS_EXTRACT_NETS` flow

  Uses the field solver to perform extraction on critical nets and standard pattern-based extraction on the remainder of the design

Table 28 summarizes the differences between the flows. You can customize the field solver extraction by using the `FSCOMPARE_OPTIONS` command, which applies to both of the field solver flows.

Table 28 Comparison of Field Solver Flows

	FSCOMPARE flow	**FS_EXTRACT_NETS flow**
Command to invoke flow	`EXTRACTION: FSCOMPARE`	`FS_EXTRACT_NETS`
Default convergence goal set by the `FSCOMPARE_OPTIONS -perc_self` command	0.5 percent	1.5 percent

Feedback

Table 28 Comparison of Field Solver Flows (Continued)

	FSCOMPARE flow	FS_EXTRACT_NETS flow
Output files	Generates the .fs_comptot and .fs_compcoup output files, which report the differences between the StarRC pattern-based extraction results and the random-walk extraction results of the field solver	Incorporates the extracted capacitances into the resulting netlist; does not generate .fs_comptot and .fs_compcoup report files

Output Files for the FSCOMPARE Flow

The field solver generates a report with the extension .fs_comptot for general capacitances and another report with the extension .fs_compcoup for coupling capacitances. The nets included in these reports are filtered by the values of the `FSCOMPARE_THRESHOLD` command for general capacitances and the `FSCOMPARE_COUPLING_THRESHOLD` command for coupling capacitances.

The .fs_comptot file contains a header with statistics for all unshorted nets, followed by three tables of nets:

- The first table is a report of unshorted nets and their total capacitance values obtained from the field solver (labeled FS) and StarRC pattern-based extraction (labeled xTract). For field solver capacitances, the percentages in parentheses are the statistical uncertainty values. The nets appear in order of the absolute value of the per-net correlation reported in the first column, from largest to smallest.
- The second table is a list of shorted nets. The short type `Drawn` indicates that the short was found by the StarRC extraction and not the field solver extraction. The file also indicates if the limit set by the `SHORTS_LIMIT` command was reached and notes the name of the file that contains detailed information about the shorts.
- The third table is a list of nets with SMIN violations. This table appears only if the `REPORT_SMIN_VIOLATION` command is set to `YES` (the default is `NO`). The file also indicates if the limit set by the `SMIN_LIMIT` command was reached and notes the name of the file that contains detailed information about the violations.

The .fs_compcoup is similar; however, it does not include a header with statistics and the first table is a report of coupling capacitances instead of total capacitances.

An example of a *.fs_comptot file is as follows:

```
StarXtract Version:
Field Solver Version:

Min Error          : 0.327769%
```

Feedback

```
Max Error          : 2.32324%
...

Detailed Report
%Diff   AbsError(fF)      FS(fF)         xTract(fF)  NetName
-------------------------------------------------------------------------
 0.55%   0.02676       4.89356 (1.2%)     4.92032     D23
-0.49%   0.0566781    11.5645  (0.5%)    11.5078      D7
 0.24%   0.0278593    11.551   (0.5%)    11.5789      S7
-0.18%   0.011445      6.49725 (1.0%)     6.4858      D22
...

Detailed Report for Shorted Nets
%Diff   AbsError(fF)      FS(fF)         xTract(fF)  ShortType  NetName
-------------------------------------------------------------------------
 91.17%   0.417215     0.45763 (0.7%)    0.87485     Drawn     UNSIG_2
-19.54%   0.238049     1.21858 (0.5%)    0.980529    Drawn     d
-12.68%   0.373508     2.94479 (0.3%)    2.57128     Drawn     vss
...
Details about drawn shorts in shorts_all.sum file.

Detailed Report for Smin Violations Nets
%Diff   AbsError(fF)      FS(fF)         xTract(fF)  NetName
-------------------------------------------------------------------------
-14.06%   24.4216      173.668 (0.3%)   149.247     RIGHT1
-14.04%   33.5042      238.646 (0.2%)   205.142     RIGHT0
-13.95%   33.2873      238.684 (0.2%)   205.397     SIGNALNET
Details about smin violations in smin.sum file.
```

Controlling Field Solver Accuracy and Runtime

In the field solver, the runtime has an inverse quadratic dependency on the accuracy. For example, reducing the accuracy tolerance by a factor of three (such as, from 3 percent to 1 percent) generally results in a ninefold increase in CPU time. The default accuracy tolerance in the field solver is 1 sigma at 0.5 percent for total capacitance. This means that the error incidence in total capacitance for 68 percent of the nets extracted is less than 0.5 percent. You can change the default 0.5 percent for total capacitance convergence to a different value using the `-perc_self` option with the `FSCOMPARE_OPTIONS` statement of the StarRC command file.

For example, to specify 1 percent as the requirement for total capacitance convergence, use the following command in the StarRC command file:

```
FSCOMPARE_OPTIONS: -perc_self 1
```

In general, the runtime is proportional to the number of nets extracted. The runtime is also dependent on the overall size of the layout (number of boxes or polygons). Increasing the number of dielectric layers can also cause the runtime to increase because the number of hops required for a walk to reach a conductor increases.

Feedback

Specifying Convergence Goals

A net is considered to be convergent if it satisfies the following relations:

Ec < perc_self x C & for each Cij { Eij < abs_coup + Cij x perc_coup}

In this equation,

- Ec = estimated statistical error on the total net capacitance
- C = total net capacitance
- Cij = coupling capacitance from net i to a neighbor net j
- Eij = estimated statistical error on Cij
- Default of abs_coup = C x coup_cap_thresh

Small coupling capacitors can be very slow to converge. To determine the value of a coupling capacitor between two electrodes A and B, a walk must start on electrode A and end on electrode B. The smaller the coupling capacitor, the smaller the probability that this happens, and a larger number of total walks is needed to obtain accurate statistics on the coupling capacitor.

For example, if the total capacitance at a node is 1e-15 farads and a coupling capacitor to the same node is 1e-18 farads, then 1000 more walks are needed to calculate the coupling capacitance value. Therefore, extracting the coupling capacitor to the same accuracy as the net total capacitance takes 1000 times longer. Usually the largest capacitances are the most important and converge the fastest in the field solver.

By default, the field solver does not check the percentage convergence of coupling capacitance. However, you can force the field solver to check the percentage convergence of coupling capacitance by using the `-perc_coup` option.

For example, to set the convergence of coupling capacitance to 10 percent, use the following command in the StarRC command file:

```
FSCOMPARE_OPTIONS: -perc_coup 10
```

By default, the field solver sets the threshold for convergence of coupling capacitance to 1 percent of the total net capacitance. To change the default, use the `-coup_cap_thresh` option.

For example, to set the absolute convergence of coupling capacitance to 2 percent of the total net capacitance, use the following command in the command file:

```
FSCOMPARE_OPTIONS: -coup_cap_thresh 2
```

Typically, it is not necessary to set coupling capacitance goals. If you do need to set them, make sure you fully understand the convergence criteria because incorrectly setting `-perc_coup` and `-abs_coup` can result in unnecessarily long runtimes.

There is no way to force the field solver to evaluate a specific coupling capacitor. Coupling capacitors are evaluated in random order, based on where the random walks start and end.

Specifying the Accuracy Goal

You can specify the accuracy of the extracted total capacitance without having to calculate the convergence level in terms of the standard deviation. In this case, the tool automatically chooses the appropriate convergence tolerance.

To specify convergence in this manner, you can set the `-perc_accuracy` and the `-perc_accuracy_confidence` options. The default of the `-perc_accuracy_confidence` option is 99.7 percent. For example, if you set `-perc_accuracy` to 5 percent and leave `-perc_accuracy_confidence` at its default, the extracted total capacitance value is accurate to 5 percent with a confidence level of 99.7 percent. The 99.7 percent confidence interval about the mean of a Gaussian distribution is 3 sigma. This is equivalent to setting `-perc_self 1.67` (that is, the standard deviation is 5 percent ÷ 3 = 1.67 percent).

Specifying the Consistency of Results

In a random walk field solver, each net has a statistical error associated with the result. If a design contains 1000 identical nets, then the extracted values vary somewhat because of this error. The statistical error in the field solver follows a normal distribution. The standard deviation of the distribution sigma (σ) is controlled by the `-perc_self` option.

For example, because the computed capacitance values follow the normal distribution, 68 percent of the nets lie within one sigma of the correct value (μ); that is, they occur between μ–σ and μ+σ, or within 2 sigma of each other. Thus, the number of nets consistent to within 2 sigma is 68 percent.

As an alternate way of setting the consistency, you can use the `-perc_consistency` and `-perc_of_nets_consistent` options. Use these options to calculate a value for the `-perc_self` option.

For example, to specify that 99.7 percent of the identical nets must have errors less than 1 percent, you can use either of the following equivalent commands:

- `FSCOMPARE_OPTIONS: -perc_consistency 1 -perc_of_nets_consistent 99.7`
- `FSCOMPARE_OPTIONS: -perc_self 0.167`

Feedback

Specifying Pattern Matching for Symmetric Nets

Memory designs often contain thousands of identical or symmetric nets with the same capacitance characteristics. The field solver uses a pattern matching process to identify these nets, analyze only a single representative net, and copy the capacitance characteristics to all matching nets. Using pattern matching can speed up the analysis of memory designs as well as ensure well-matched capacitance values for identical or symmetric nets.

To enable pattern matching, specify the `-match` option in the `FSCOMPARE_OPTIONS` command.

Distributed Processing for Field Solver Jobs

During distributed processing, the field solver distributes the nets to be extracted among the available processors. One StarRC license enables up to four field solver distributed processing jobs

Specify the number of processors with the `FSCOMPARE_OPTIONS: -np` command and the number of threads per processor with the `FSCOMPARE_OPTIONS: -nt` command. If you specify both commands, the same number of threads is used on each processor.

For example, the following statement specifies to use four processors:

```
FSCOMPARE_OPTIONS: -np 4
```

The following statement specifies to use two threads on each of two processors:

```
FSCOMPARE_OPTIONS: -np 2 -nt 2
```

If the total number of specified threads (the product of the number of processors and the number of threads per processor) exceeds the number of nets to be extracted, the field solver uses only the same number of threads as nets to be extracted to avoid using unnecessary threads. Using distributed processing on small jobs that would only take a few minutes does not typically improve runtime because of the startup time for the machines.

Distributed processing of the field solver can tolerate machine failures. If a field solver job on one machine or thread is lost during a run, the lost job is taken over by the remaining available machines or threads to ensure that the run is successful.

Setting Up Distributed Processing

The job submission command can be specified either in an environment variable or as a command in the StarRC command file. If set in both places, the StarRC command file

takes precedence. The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Distributed processing is available for the following computing environments:

- LSF System
- Univa Grid Engine
- General Network With a List of Machines
- Runtime Design Automation System

LSF System

The submission command for LSF systems is `bsub`. Specify it with one of these methods:

- Using the `STARRC_DP_STRING` environment variable, which must be set before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example:

```
% setenv FS_DP_STRING 'bsub -R "rusage[mem=5000]"'
```

- Using the `FS_DP_STRING` command in the StarRC command file. For example:

```
FS_DP_STRING: bsub -R "rusage[mem=5000]"
```

Univa Grid Engine

The submission command for Univa Grid Engine systems (formerly known as Oracle Grid Engine) is `qsub`. Specify it with one of these methods:

- Using the `STARRC_DP_STRING` environment variable, which must be set before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example:

```
% setenv FS_DP_STRING 'qsub -P bnormal -l  "mem_free=1G" \
  -v "STARRC_LICENSE_WAIT=YES"'
```

- Using the `FS_DP_STRING` command in the StarRC command file. For example:

```
FS_DP_STRING: qsub -P bnormal -l "mem_free=1G" \
  -v "STARRC_LICENSE_WAIT=YES"
```

To pass environment variables to the job, you must use the `-v` option. In this example, the `STARRC_LICENSE_WAIT` environment variable is set to allow license queuing.

General Network With a List of Machines

For a general network with a list of machines, the syntax is

```
FS_DP_STRING: list host1[:n1] [host2[:n2] ... hostm[:nm]]
```

where `host1:n1` means that you are submitting n1 jobs (an integer number of jobs) to the machine with name host1. The default number of jobs per machine is 1. Do not use spaces inside the *`host:n`* syntax; use one or more spaces between host machines. Each machine must be available through a simple UNIX `rsh` command without a password.

The keyword for the machine where the job is started is `localhost`. If you use `localhost`, system calls are used instead of rsh to submit the jobs.

You can specify the `FS_DP_STRING` variable in either of the following forms:

- Using the `STARRC_DP_STRING` environment variable, which must be set before launching the tool. Enclose the list in single quotation marks because it might contain multiple arguments. For example:

  ```
  % setenv FS_DP_STRING 'list alpha beta gamma'
  ```

- Using the `FS_DP_STRING` command in the StarRC command file. For example:

  ```
  FS_DP_STRING: list alpha:1 beta:4 gamma:2
  ```

Runtime Design Automation System

The submission command for a Runtime Design Automation (RTDA) system is `nc run`. Specify it with one of these methods:

- Using the `STARRC_DP_STRING` environment variable, which must be set before launching the tool. Enclose the entire command in single quotation marks because it might contain multiple arguments. For example:

  ```
  % setenv FS_DP_STRING 'nc run -R "rusage[mem=5000]"'
  ```

- Using the `FS_DP_STRING` command in the StarRC command file. For example:

  ```
  FS_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
  ```

7

Using StarRC With the Custom Compiler Tool

The Synopsys Custom Compiler tool is a custom design environment that unifies design, simulation, layout, physical verification, parasitic extraction, and static timing analysis. Running the StarRC tool to perform extraction within the Custom Compiler environment provides features such as interactive analysis and visualization.

This user guide provides a general overview of the Custom Compiler tool. However, for up-to-date information about the features and user interface, see the Custom Compiler documentation on SolvNetPlus.

For more information, see the following topics:

Extraction Features in the Custom Compiler Tool

The Custom Compiler design environment uses the StarRC tool to extract and investigate parasitic capacitance for interconnect layers. The Custom Compiler tool provides a graphical user interface (GUI) that helps you to set up and execute most of the available operations. The extraction features are as follows:

- Parasitic View Generation

  Set up and execute a StarRC extraction run within the Custom Compiler environment. Read schematic and layout views, then generate parasitic views.

- In-Design Extraction

  Investigate coupling capacitance between partially or fully routed nets. Visualize capacitances with crossprobing capability between the layout and the report.

- Estimated Parasitics Assistant

  Calculate parasitic capacitance interactively during the design process.

All extraction operations require similar prerequisite files, which are described in the following topics:

- Overview of the Custom Compiler Extraction Flow
- The Device Mapping File
- The Layer Mapping File

You can also look at the following references:

- *Custom Compiler Schematic Editor and Text Editor User Guide*
- *Custom Compiler Electrical Checking and Reporting User Guide*
- *Custom Compiler Design Checking and Physical Verification User Guide*

Parasitic View Generation

You can set up and execute StarRC extraction within the Custom Compiler environment.

As a prerequisite, you must obtain the following files from the foundry:

- The nxtgrd file, which contains process modeling information
- The LVS runset for the LVS tool in your flow
- The StarRC command file (called a runset in the Custom Compiler interface)

Feedback

Before performing extraction, you must perform LVS analysis using one of the following tools:

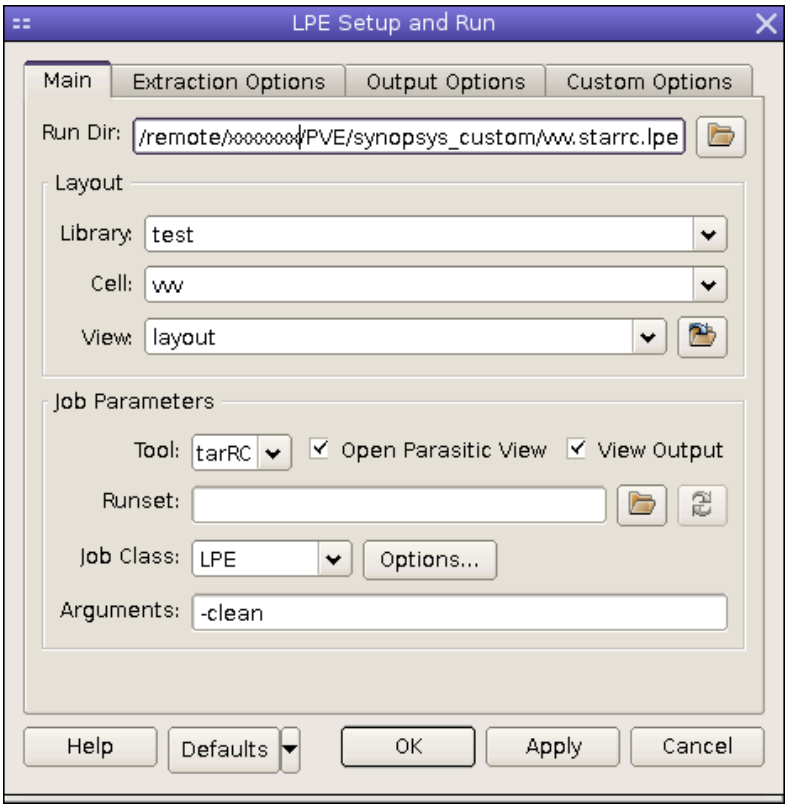
- Synopsys IC Validator
- Synopsys Hercules
- Mentor Graphics Calibre

Extraction Setup

The Custom Compiler extraction user interface, which is titled LPE Setup and Run, has four tabs for setup information.

For example, the Main tab is shown in Figure 38. In this tab, you specify the run directory, the layout to extract, and the StarRC command file (in the Runset field). You can optionally set up distributed processing and other job control options by using the Options button and the Job Class field. The Arguments field specifies options for the `StarXtract` command.

Figure 38 Custom Compiler Extraction Setup Main Tab



The other tabs are as follows. See the *Custom Compiler Design Checking and Physical Verification User Guide* for screen shots and detailed information.

- The Extraction Options tab

  On the Extraction Options tab, you must specify the runset report file, which is created by the LVS tool and which is necessary for input to the StarRC tool. In addition, this tab allows you to specify options such as the corner of interest, the operating temperature, the extraction type, and the amount of reduction to perform.

- The Output Options tab

  On the Output Options tab, you must specify the mapping file names. You can optionally provide library, cell, and view names to use for port and property annotation. In addition, you can select specific nets for extraction.

  The output runset field specifies the name of a file in which to save the final StarRC commands. The output runset file includes the commands read from the command file (runset file) specified on the Main tab followed by StarRC commands that reflect the settings specified in the Custom Compiler user interface.

  Duplication of StarRC commands is treated the same as in any StarRC command file. The last command in the file takes precedence over earlier instances of the same command, with the exception of certain commands that specify lists of objects, such the `NETS` and `NETLIST_SELECT_NETS` commands, which are additive. See the command reference pages in Chapter 14, StarRC Commands for specific command behavior.

- The Custom Options tab

  The Custom Options tab allows you to specify StarRC commands that are not included in the GUI.

After you complete the setup, click OK in the LPE Setup and Run interface to initiate the extraction. Alternatively, you can use these setup conditions to control one-button extraction performed from the layout editor.

Extraction Results

After extraction is complete, analysis and visualization features include the following:

- Annotate capacitors to the schematic for easy visualization
- Use the CrossTalk Checker feature to investigate coupling capacitances
- Create customized reports
- Cross-probe between the schematic view and parasitics view

Feedback

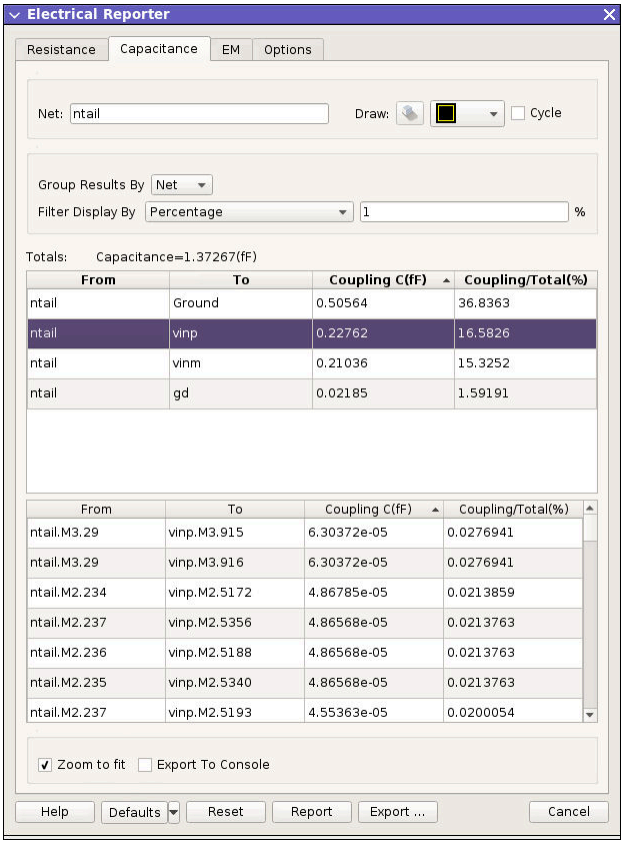
In-Design Extraction

The Capacitance Reporter in the Custom Compiler schematic editor uses the StarRC tool to generate interactive reports of the capacitances in partially completed designs.

You must first set up capacitance extraction by specifying the nxtgrd file (process modeling information) and the layer mapping file in the Options tab of the Electrical Reporter assistant. You then right-click the net of interest in the layout editor Design Navigator and select Report Capacitance from the menu.

Figure 39 is an example of the Capacitance Reporter output. The display includes the total net capacitance and the net length. The upper table lists all coupling capacitances to other nets. If you click one of the coupling capacitance lines in the upper table, the Capacitance Reporter provides detailed information about that specific capacitance in an additional table at the bottom of the display.

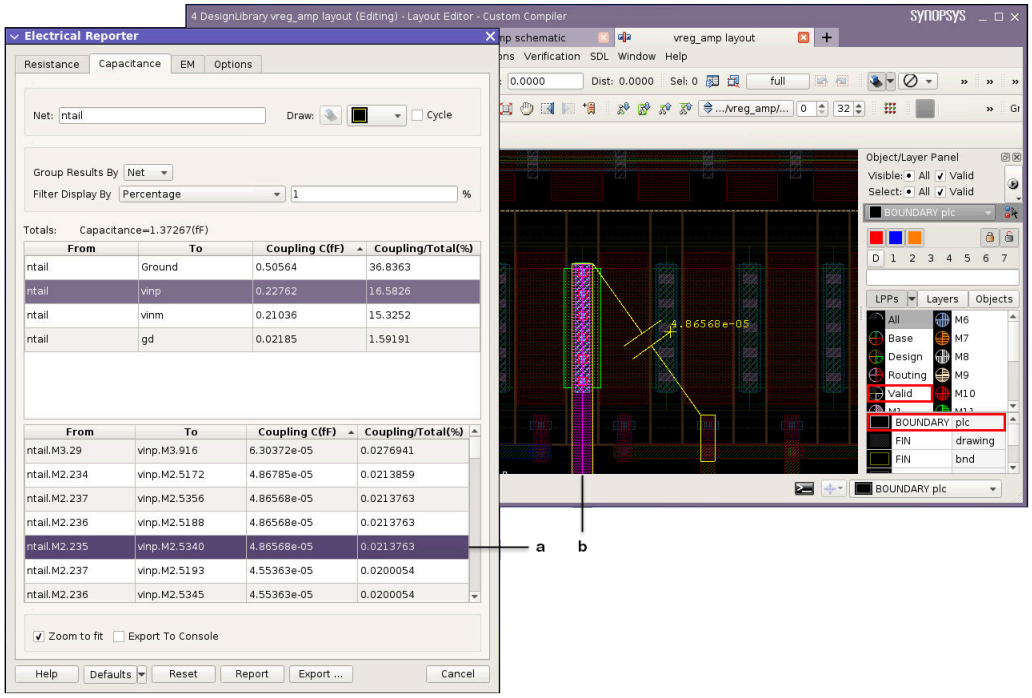
Figure 39 Capacitance Reporter Output



Feedback

The schematic editor canvas interacts with the Capacitance Reporter by displaying the location and value of any capacitance that you select in the Capacitance Reporter tables, as shown in Figure 40.

Figure 40 Capacitance Reporter Interaction With Schematic



Estimated Parasitics Assistant

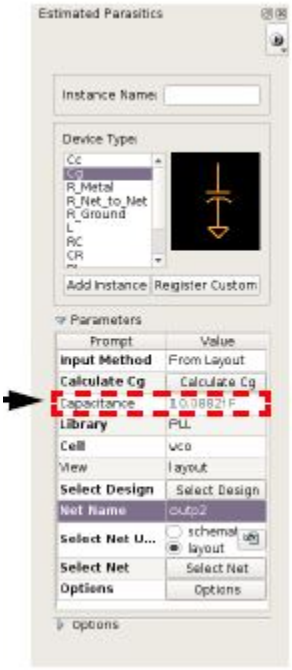
The Estimated Parasitics assistant in the Custom Compiler schematic editor allows you to add estimated or calculated capacitance to a design for more realistic simulation. You can provide your own estimate of a capacitance value for a net, or you can use the StarRC tool to calculate capacitance from the wire geometry in the layout.

To perform capacitance calculations from the layout, you must first set up the StarRC options, such as the nxtgrd file and layer mapping file, in the Options section of the Estimated Parasitics Assistant. You can select the net of interest from either the schematic or the layout.

Finally, select Calculate Cg to run the StarRC tool in the background. After the calculated capacitance is displayed, you can add the parasitic device instance to the design.

Feedback

Figure 41 Estimated Parasitics Assistant



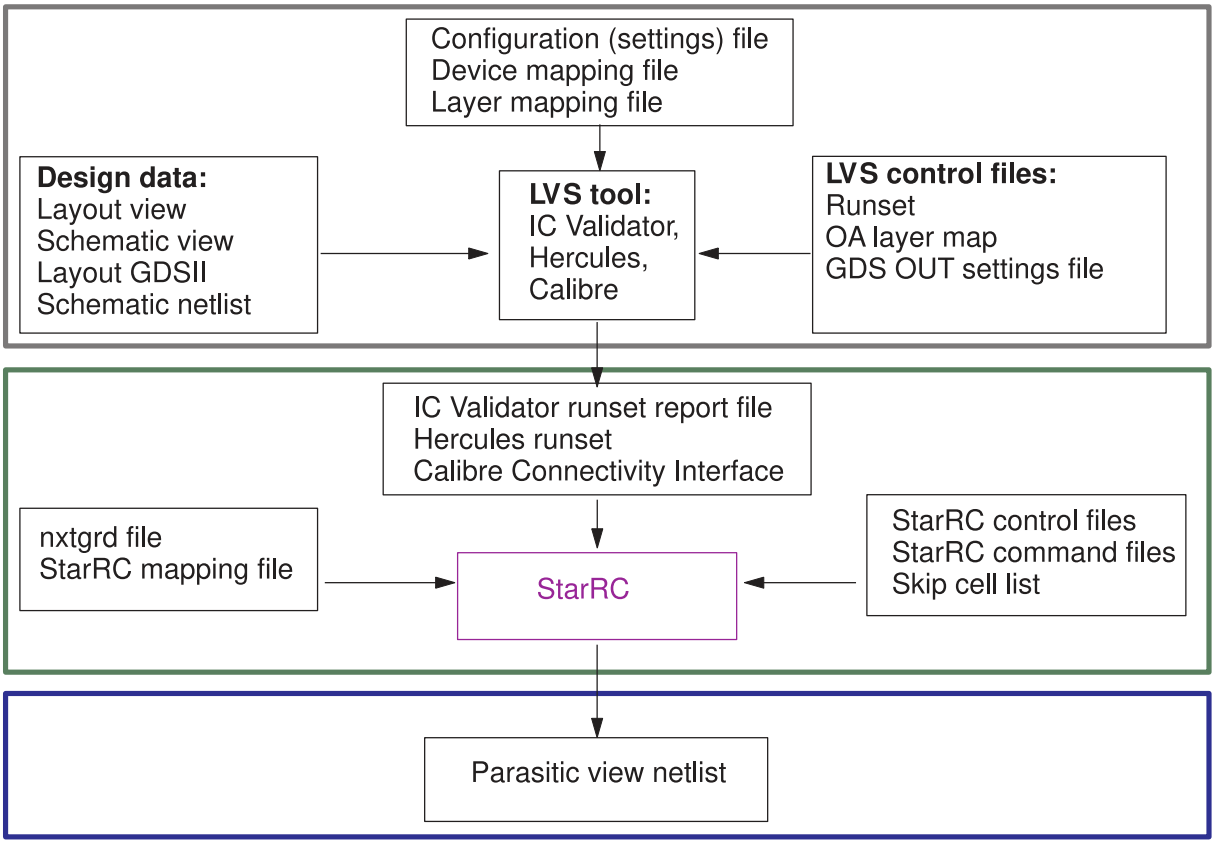
Feedback

Overview of the Custom Compiler Extraction Flow

The Custom Compiler extraction flow has three stages, as shown in Figure 42:

- Layout versus schematic (LVS) stage, using the IC Validator, Hercules, or Calibre tools
- Extraction stage, in which you specify StarRC options
- Output stage, in which you select the output format and other options

Figure 42 Parasitic Extraction Flow in the Custom Compiler Flow



License Requirements

Using extraction features in the Custom Compiler tool requires that the StarRC tool be installed. You must have the appropriate StarRC licenses for the StarRC commands used in the extraction.

Feedback

The Device Mapping File

The device mapping file maps ideal and parasitic devices in the StarRC parasitic output to the corresponding device symbols in the Custom Compiler symbol libraries. This file contains an entry for every ideal and parasitic device model that exists in the parasitic output. It also provides the ability to map standard StarRC DSPF device property names to user-specified property names. The format of a single line must use the following format:

```
RC_model_name oa_lib_name oa_cell_name oa_view_name
     oa_symbol_pin_1 oa_symbol_pin_2 [oa_symbol_pin_3] …
     PROPMAP DSPFprop1 = CCprop1 ...
```

Two forward slashes (//) serve as a comment delimiter in the device mapping file.

Table 29 lists the arguments and their definitions.

Table 29 Device Mapping File Arguments

Argument	Definition
RC_model_name	StarRC output model name; corresponds to the schematic device model name when XREF is activated. Note that keywords pres and pcap are used for parasitic resistors and capacitors.
oa_lib_name	Name of Custom Compiler library containing the corresponding device symbol.
oa_cell_name	Cell name of Custom Compiler device symbol.
oa_view_name	View name of Custom Compiler device symbol.
oa_symbol_pin_N	Name of the pin inside Custom Compiler device symbol that corresponds to terminal #N in an analogous DSPF-based StarRC parasitic output. Note that the ordering of Custom Compiler symbol pin names inside the device mapping file must match the StarRC netlist pin order for the device type of interest. The keyword `nil` can be specified for any *oa_symbol_pin* to indicate that the terminal #N in the StarRC parasitic output should be ignored when connecting the corresponding Custom Compiler library symbol.
DSPFpropN = CCpropN	Optional mapping of standard DSPF property name into a user-specified property name. Note that keyword PROPMAP is required before the first property name mapping entry. Setting *DSPFpropN* `= nil` prevents the listed property from being annotated to the device symbol.

An example symbol mapping file is as follows:

```
MNM  devlib  nmos4  ivpcell  D G S B PROPMAP l=simL w=simW
MPM  devlib  pmos4  ivpcell  D G S B PROPMAP l=simL w=simW
pres Lib  presistor  auLvs PLUS MINUS
pcap Lib  pcapacitor auLvs PLUS MINUS
```

Feedback

```
pind analogLib pinductor symbol PLUS MINUS
pmind analogLib pmind symbol
```

Parasitic elements pres, pcap, pind, pmind should use the lib/cell/view from analogLib. For example:

```
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
pind analogLib pinductor symbol PLUS MINUS
pmind analogLib pmind symbol
```

However, if a parasitic element name conflicts with a user-defined device name, the Custom Compiler tool provides the following parasitic element names:

- `pres[starrc]`
- `pcap[starrc]`
- `pind[starrc]`
- `pmind[starrc]`

When `pres` and `pres[starrc]` both appear in the device mapping file, `pres[starrc]` overrides `pres` as the parasitic element name.

The Layer Mapping File

The layer mapping file maps layers in the StarRC mapping file to the corresponding Custom Compiler technology file layers. This allows polygons, ports, and subnodes from the parasitic extraction to be stored within the generated OA parasitic view.

Each layer that you want to store in the parasitic view should be specified in the layer mapping file and should be mapped to an existing layer-purpose pair (LPP) from the Custom Compiler technology library for the library being used. The mapping file optionally lets you specify OA layer-purpose pairs for subnode markers generated by the StarRC tool to represent parasitic top-level ports (*|P), instance ports (*|I), and subnodes (*|S).

The format of the layer mapping file is as follows:

```
RC_MAPPING_FILE_layer
   oa_polygon_layer_name oa_polygon_purpose_name
   [ oa_port_layer_name oa_port_purpose_name
   [ oa_subnode_layer_name oa_subnode_purpose_name]]

RC_MAPPING_FILE_layer
   oa_polygon_layer_name oa_polygon_purpose_name
   [ oa_port_layer_name oa_port_purpose_name
   [ oa_subnode_layer_name oa_subnode_purpose_name]]
. . .
```

Feedback

Two forward slashes (//) serve as a comment delimiter in the device mapping file.

Argument	Definition
RC_MAPPING_FILE_layer	Database layer name from the CONDUCTING_LAYERS, VIA_LAYERS, or MARKER_LAYERS sections of the mapping file specified by the MAPPING_FILE command.
oa_polygon_layer_name *oa_polygon_purpose_name*	Layer name and purpose name from the technology library, which forms the layer-purpose pair to which RC_MAPPING_FILE_layer polygons should be written. If either entry is not specified or are specified as nil, polygons corresponding to the RC_MAPPING_FILE_layer is not generated within the parasitic view.
oa_port_layer_name *oa_port_purpose_name*	Layer name and purpose name from the technology library, which forms the layer-purpose pair to which parasitic port markers corresponding to RC_MAPPING_FILE_layer interconnect should be written. Parasitic port markers are analogous to *|P nodes and .SUBCKT header nodes that would appear in the DSPF output. If either entry is not specified or are specified as nil, ports corresponding to the RC_MAPPING_FILE_layer is not generated within the parasitic view.
oa_subnode_layer_name *oa_subnode_purpose_name*	Layer name and purpose name from the technology library, which forms the layer-purpose pair to which parasitic subnode markers corresponding to RC_MAPPING_FILE_layer should be written. Parasitic subnode markers are analogous to *|I and *|S nodes that would appear in a DSPF output. If not specified or if specified as nil, subnodes corresponding to the RC_MAPPING_FILE_layer is not generated within the parasitic view.

Polygons are written to the parasitic view only if the LVS database layer of the polygon is mapped to a valid Custom Compiler layer-purpose pair in the layer mapping file. If an LVS database layer is not listed in the mapping file, no polygons corresponding to that layer are stored in the parasitic view. If the file is not supplied at all, no graphical data is written.

Because all ports and subnodes correspond to specific database layers in standard StarRC outputs, separate layer-purpose pairs are used in the layer mapping file for the generation of port and subnode markers relative to the generation of interconnect polygons. Port and subnode markers enable point-to-point resistance probing with the StarRC parasitic probing utility. Therefore, failure to include layer-purpose pairs for port and subnode markers prohibits the probing of point-to-point resistance between nodes lacking such markers.

A layer-purpose pair cannot be used both as a polygon LPP and a port and subnode LPP. If an LPP used as a polygon LPP is found in the mapping file, that LPP is disregarded if it is subsequently listed in the mapping file as a port or subnode LPP. Likewise, if an LPP used as a port or subnode LPP is found in the mapping file, that LPP is disregarded if it is subsequently listed in the mapping file as a polygon LPP.

The following example shows a layer mapping file:

```
M1 metal1 net metal1 port metal1 node
fpoly poly net poly port poly node
ngate poly net poly port poly node
pgate poly net poly port poly node
diff_con cc net nil nil cc node
subtie pad drawing nil nil pad node
welltie pad drawing nil nil pad node
nsd nactive net nactive port nactive node
psd pactive net nactive port pactive node
m1_pio_marker nil nil metal1 port nil nil
m2_pio_marker nil nil metal2 port nil nil
m3_pio_marker nil nil metal3 port nil nil
poly_marker nil nil poly port nil nil
```

Parasitic View Generation Conventions

Parasitic view generation is described in the following topics:

- Net and Instance Name Conventions
- Port and Terminal Connectivity Characteristics
- Instance Property Annotation From the Schematic View
- Subnode Marker and Parasitic Device Visualization

Net and Instance Name Conventions

The StarRC parasitic view abides by naming conventions for nets and instances to enforce uniformity with Custom Compiler naming rules and naming conventions for schematic-based parasitic simulation analysis. These naming conventions are unique to the Custom Compiler flow and are different from standard StarRC DSPF netlist conventions.

The pipe character (|) serves as the default hierarchical delimiter for the parasitic view.

During schematic view netlist creation for LVS and StarRC schematic-based cross-referencing (using the StarRC `XREF` command), the schematic view netlist generator can append prefixes to instance names according to the conventions of the netlist generator. These prefixes, commonly called SPICE cards, propagate into the StarRC parasitic netlist when the `XREF` command is used. However, these extra instance prefixes are not present in the original schematic view and might impede the ability to perform full parasitic-view to

schematic-view matching during simulation. Therefore, the Custom Compiler flow removes these instance name prefixes as follows:

- Strips the initial SPICE card from ideal (not parasitic) instance names, because the StarRC tool adds this card directly.
- For hierarchical instance names, including those preceding internal net names:
  - Decomposes the name according to the hierarchical delimiter.
  - Strips the initial X character if a decomposed instance name begins with X anywhere in the decomposed instance list.
  - If the last name in the decomposed instance (often a primitive) begins with two occurrences of the same character, strips one of them.

Table 30 Examples of Removal of Instance Name Prefixes

StarRC DSPF instance name		Parasitic view instance name
XI0/XI5/M1	>	I0/I5/1
XI0/XI5/MM1	>	I0/I5/1
I0/I5/M1	>	I0/I5/M1
I0/I5/MM1	>	I0/I5/M1
XI0/X15/net1	>	I0/I5/net1
I0/I5/net1	>	I0/I5/net1

- If the last name in the decomposed instance (often a primitive) begins with a SPICE character such as X or M, strips that character.

The naming convention for input data makes the following assumptions:

- The schematic netlist generator always appends an X to nonprimitive cell instances (for example, instances in the middle of a flattened hierarchical name).
- No instance name inside the schematic view begins with X.
- No instance name inside the schematic view begins with two occurrences of the same letter, such as the schematic view instance MM0.

Occurrences of bus bits (name<#>) are renamed if the bus bit is embedded within the middle of a hierarchical instance name. In such cases, the embedded string <#> is replaced with the embedded string (#). An example where this behavior can occur is an iterated schematic instance name embedded in a hierarchical name. For example, `[I0|I1|I2<3>|net4` becomes `I0|I1|I2(3)|net4`.

Feedback

Port and Terminal Connectivity Characteristics

The StarRC tool reads the following information from a pre-existing view of the extracted cell and populates the same information within the parasitic view:

- netExpression parameters

  The tool parses all terminals and signals in the ports global nets view (default is layout) and records any netExpression parameters. If a terminal and a signal have the same name and both have individual netExpressions, only the netExpression of the terminal is recorded. The netExpressions are then propagated into the new parasitic view as follows:

  - If a terminal in the parasitic view has a name matching a terminal or signal name for which a netExpression was read from the existing cell, the netExpression is added to that terminal.
  - Otherwise, if a signal in the parasitic view has a name matching a cached terminal or signal name for which a netExpression was read from the existing view, the netExpression is added to that signal.

  Note:

  Terminals in the parasitic view are dictated by the presence of ports in the StarRC parasitic output which are analogous to *|P nodes or .SUBCKT headers in a DSPF netlist. If no such nodes exist, the tool does not create any terminals in the parasitic view and no netExpression parameters are transferred.

- Direction parameters for terminals

  The StarRC tool parses all terminals in the pre-existing view and records any direction parameters listed on any terminals. If a terminal in the parasitic view has the same name as a terminal with a direction parameter in the pre-existing view, the tool attaches the same direction parameter string to the matching terminal in the parasitic view.

- isGlobal parameters for signals

  The tool parses all signals in the pre-existing view and records any isGlobal parameters listed on any signals. If a signal in the parasitic view has the same name as a signal with an isGlobal parameter in the pre-existing view, the tool attaches the same isGlobal parameter string to the matching terminal in the parasitic view. Note that the isGlobal parameter is propagated only for signals to which no terminals bearing netExpression parameters are attached. The pre-existing view from which the previous connectivity and terminal information is read is specified by the corresponding field in the Custom Compiler user interface.

  When updating a terminal view, the StarRC tool gathers terminal information from a given symbolic view and parses all signals in the parasitic view to check for floating

nets on device instance terminals. If a net name matches one terminal name on the symbol view, the tool creates the name as the terminal name for the parasitic view.

- Power net name for the ports

  The StarRC tool creates the ports of the parasitic view as the input database top-block port names. Some additional power pins might be necessary for the parasitic view to connect to upper-level views.

Instance Property Annotation From the Schematic View

Properties on the instance inside the parasitic views come from one of two places, either the schematic view or the LVS tool. There might also be custom usages, such as to copy a property value to multiple property names, delete a property, change a property name, or create a new property. The StarRC tool needs to set the property names and values in the parasitic view for the simulator to work correctly.

Property Mapping

This section describes syntax to adjust property annotation behaviors. These behaviors are applied to all properties, both in the schematic view and LVS results.

Table 31 Settings for Property Mapping

Syntax	Description
`dspfProp`	Property name or value in the StarRC DSPF results, usually from LVS results
`schProp`	Property name or value In the schematic view
`cdfProp`	Property name defined in the CDF file
`preProp`	Property name or value before StarRC parasitic view generation
`paraProp`	Property name or value in the final parasitic view

Property Mapping Behavior

The following examples describe property mapping operations. Use StarRC commands such as the `OA_PROPMAP_CASE_SENSITIVE` and `OA_PROPERTY_ANNOTATION_VIEW` commands to control property mapping.

A=B

In the instance of the parasitic view, paraProp B gets its value from preProp A. Also, paraProp A value comes from preProp A. This is equivalent to A=A and A=B.

A=B and A=C

Similar to A=B, but with multiple usage.

A>B

Two actions occur: paraProp B gets its value from preProp A and paraProp A is removed. If A is a cdfProp and cannot be removed, the value is set to nil such that A=nil and A=B.

A="*constant*"

A constant is assigned to paraProp A, regardless of the original value. If there is no paraProp with the name A, the tool creates one with the assigned value.

A=""

Remove paraProp A; if A is a cdfProp and cannot be removed, the value of A is set to an empty string.

Instance Name Matching Rule

The StarRC tool performs instance property annotation by finding the corresponding instance in the schematic view. The tool also performs SPICE card stripping of schematic view instances, then finds the corresponding instance in the schematic view to obtain the properties for annotation.

When the tool cannot find a schematic view instance to annotate properties to parasitic view instances, the failed instances are reported in a file named schematic_info_log.

Subnode Marker and Parasitic Device Visualization

The layer mapping file provides the ability to write extracted polygon data to the parasitic view. Besides the storage of interconnect polygons, graphical data including subnode markers, port markers, and pres or pcap flylines are also written to the parasitic view. For more information about the layer mapping file, see The Layer Mapping File.

A subnode is defined as any *|I or *|S node that would normally occur in a standard StarRC DSPF parasitic netlist. A port is analogous to any *|P node or any entry in the .SUBCKT header line of a standard DSPF parasitic netlist. These nodes represent the electrical connection points for parasitic devices and ideal devices. Every subnode has unique xy coordinates as well as an extracted database layer on which the subnode lies. Using this information, it is possible to represent the subnodes in the parasitic view with small marker shapes placed at their corresponding xy coordinates.

The OA layer-purpose pair to which a port or subnode marker is written is defined in the OA layer mapping file. Only ports or subnodes whose corresponding database layers are listed in the OA layer mapping file has markers written to the parasitic view.

Feedback

Graphical data is also stored for parasitic resistors and coupling capacitors in the form of flylines between subnodes. Flylines are not stored for grounded capacitors because such capacitors by definition do not terminate at a finite xy coordinate on an interconnect polygon. These flylines are annotated with two properties: the parasitic instance name and the parasitic value. All flylines for parasitic resistors are written to a single Custom Compiler layer-purpose pair. Likewise, all flylines for parasitic capacitors are written to a separate Custom Compiler layer-purpose pair. These layer-purpose pairs can be listed in the layer mapping file as follows, using the special tags *pres and *pcap.

```
// <*pres or *pcap> oa_polygon_lyr oa_polygon_purpose
*pres poly net
*pcap metal1 net
```

If no *pres and *pcap lines are defined in the layer mapping file, the StarRC tool uses the following default mapping:

```
*pres y0 drawing
*pcap y1 drawing
```

Note:

A flyline is stored in the parasitic view only if both of its nodes have markers stored in the parasitic view. Likewise, port and subnode markers are only stored for runset layers that are mapped in the layer mapping file. Therefore, the number of parasitic resistor and capacitor flylines present in the parasitic view is a direct function of the runset layer to layer mappings in the layer mapping file.

The OpenAccess Parasitic View

The OpenAccess view is a parasitic view that can be used directly in the Custom Compiler design environment. This view includes all parasitic components and network connections along with physical polygons.

The StarRC tool can generate OpenAccess format parasitic views outside the custom design environment. However, the Custom Compiler GUI is helpful for setting up many options.

The OpenAccess Flow

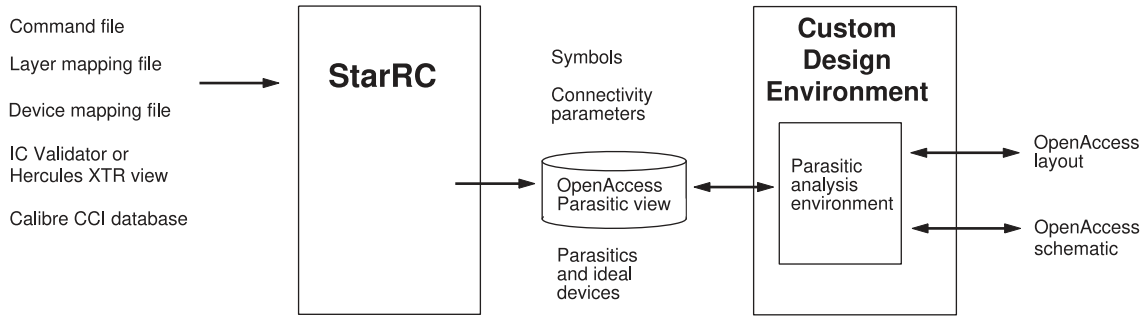
Figure 43 illustrates the OpenAccess flow in the custom design environment. You can run extraction and generate a parasitic view within the design environment for efficient post-layout simulation.

Accurate layout representation requires netlist connectivity information to instantiate each extracted parasitic element and design device. The parasitic view provides you with an efficient and detailed analysis tool in the physical domain. The parasitic view must contain

Feedback

schematic symbols that can be written in the netlist for simulation as well as used as a graphical tool for identifying parasitic devices.

Figure 43 OpenAccess Flow



Support for Special StarRC Flows

The StarRC simultaneous multicorner flow is supported for OpenAccess views.

The temperature sensitivity flow is also supported for the creation of multiple OA views.

Skip Cell Mapping

If you use skip cells in the Custom Compiler flow, the number of ports and the port names in the OpenAccess symbol view of the skip cell must match the number of ports and the port names created from the layout during layout versus schematic (LVS) checking done before parasitic extraction.

If the number of ports and the port names match, use the `OA_SKIPCELL_MAPPING_FILE` command to specify a file that defines which cell master to use for the skip cells defined in the `SKIP_CELLS` command. The syntax is as follows:

```
INV1 myLib INV symbol
```

If there is a mismatch in either the number of ports or the port names, use the following guidelines to ensure that skip cells are properly connected in the OpenAccess view created by the StarRC tool:

- The number of ports is different, but port names match

  Use the `SPICE_SUBCKT_FILE` command to specify SPICE files that contain .subckt definitions for the skip cells. The StarRC tool uses the schematic ports found in the SPICE files instead of the LVS port definitions.

  In addition, use the `OA_SKIPCELL_MAPPING_FILE` command to specify a file that defines which cell master to use for the skip cells defined in the `SKIP_CELLS` command.

Feedback

- The number of ports matches, but some or all port names are different

  Use the OA_DEVICE_MAPPING_FILE command to specify a file that contains skip cell mapping definitions. An example of the syntax is as follows:

  ```
  mimcap myLib mimcap symbol PLUS MINUS SHIELD1 SHIELD2
  ```

- The number of ports and the port names are both different

  You can define both the number of ports and the port names in the device mapping file (specified with the OA_DEVICE_MAPPING_FILE command).

  Alternatively, you can define the number of ports with SPICE .subckt files (specified with the SPICE_SUBCKT_FILE command) and the port names with a device mapping file (specified with the OA_DEVICE_MAPPING_FILE command).

OpenAccess File Examples

The following sections provide examples for the OpenAccess library definition and the layer and device mapping files.

OpenAccess Library Definition

The following is an example of the OpenAccess library specified by the OA_LIB_DEF command in the StarRC command file:

```
DEFINE w_xxb_fifo1 /remote/engr5/OA/w_xxb_fifo1
DEFINE N90lo /testcases/misc/star_customcompiler/N90lo
DEFINE snpsDefTechLib /global/apps/cc/snpsDefTechLib
DEFINE basic /global/apps/cc/basic
DEFINE snpsDeviceLib /global/apps/cc/snpsDeviceLib
DEFINE analogLib /global/apps/cc/analogLib
DEFINE verilogaLib /global/apps/cc/verilogaLib
DEFINE sheets /global/apps/cc/sheets
DEFINE sample /global/apps/cc/sample
DEFINE parasitics /global/apps/cc/parasitics
```

OpenAccess Mapping Files

The following is an example of the OpenAccess layer mapping file specified by the StarRC OA_LAYER_MAPPING_FILE command:

```
poly     PO drawing PO pin PO dummy
metal1   M1 drawing M1 pin M1 dummy
metal2   M2 drawing M2 pin M2 dummy
metal3   M3 drawing M3 pin M3 dummy
metal4   M4 drawing M4 pin M4 dummy
metal5   M5 drawing M5 pin M5 dummy
metal6   M6 drawing M6 pin M6 dummy
metal7   M7 drawing M7 pin M7 dummy
Cont  CO drawing nil nil nil nil
```

Feedback

```
pl3co   CO drawing nil nil nil nil
VIA1    VIA1 drawing nil nil nil nil
VIA2    VIA2 drawing nil nil nil nil
VIA3    VIA3 drawing nil nil nil nil
VIA4    VIA4 drawing nil nil nil nil
VIA5    VIA5 drawing nil nil nil nil
VIA6    VIA6 drawing nil nil nil nil
```

The following is an example of the OpenAccess device mapping file specified by the StarRC `OA_DEVICE_MAPPING_FILE` command:

```
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
nch N90lo nch auLvs G S D B
pch N90lo pch auLvs G S D B
nch_18 N90lo nch_18 auLvs G S D B
```

StarRC Commands for OpenAccess Parasitic Views

The StarRC commands that affect the OpenAccess flow are listed in Table 32.

Table 32 Commands for OpenAccess flow

Command	Type	Default	Description
`NETLIST_FORMAT`	string	`NETNAME`	Set to `OA`; required
`OA_BUS_BIT`	string	Same as `BUS_BIT`	Specifies the bus bit delimiter
`OA_CDLOUT_RUNDIR`	string	none	Directory that contains the ihnl subdirectory and mapping files
`OA_CELL_NAME`	string	none	OpenAccess cell name
`OA_DEVICE_MAPPING_FILE`	string	none	File that describes device mapping
`OA_INSTANCE_PIN_NAME`	string	`SUBCKT`	Specifies where to get pin names
`OA_LAYER_MAPPING_FILE`	string	none	File that describes layer mapping
`OA_LIB_DEF`	string	lib.defs	OpenAccess library definition file; optional
`OA_LIB_NAME`	string		OpenAccess library name
`OA_MARKER_SIZE`	float	0.1	Port or subnode marker size (microns); optional
`OA_MULTI_OUTPUT`	string	SPF	Format of parasitic netlist to generate in addition to OA parasitic view

Feedback

Table 32 Commands for OpenAccess flow (Continued)

Command	Type	Default	Description
OA_OVERWRITE_LOCKED_VIEW	Boolean	NO	Allows the StarRC tool to overwrite a locked parasitic view
OA_PORT_ANNOTATION_VIEW	string	null string	Enables the simulation of a parasitic view; generated by the OpenAccess writer
OA_PROPERTY_ANNOTATION_VIEW	string	none	Specifies which schematic library, cell, or view is used to check against ideal devices for schematic-only properties and to attach them into the OpenAccess parasitic view
OA_PROPMAP_CASE_SENSITIVE	Boolean	NO	Case sensitivity of property mapping
OA_REMOVE_DUPLICATE_PORTS	Boolean	NO	Prevents duplication of port names
OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX	Boolean	YES	Specifies whether to remove SPICE card prefix characters before primitives in ideal instance names
OA_REMOVE_SPICECARD_PREFIX	Boolean	YES	Specifies whether to remove SPICE card prefix characters from instance name paths
OA_SKIPCELL_MAPPING_FILE	string	none	Specifies cell master to use for skip cells in the parasitic view
OA_VIEW_NAME	string	starrc	Parasitic view name

8

Using StarRC With the Virtuoso Tool

This chapter includes the following topics that describe the interface with the Cadence® Virtuoso® custom design platform.

For more information, see the following topics:

Feedback

Introduction to Virtuoso Integration

The Virtuoso Integration (VI) interface allows you to use the StarRC tool to extract and investigate parasitics within the Virtuoso Analog Design Environment. The primary features are as follows:

- Parasitic View Generation—Set up and execute a StarRC extraction run within the Virtuoso environment. Read schematic and layout views, then generate parasitic views.

  Parasitic view creation entails the generation of a DFII CDBA database or OpenAccess parasitic view that instantiates all ideal and parasitic devices extracted by the IC Validator, Hercules, or Calibre flows. This view is compatible with common netlist generation interfaces.

- Parasitic Prober—Examine the relationships between design elements and extracted parasitics.

  Built-in highlighting capabilities enable you to highlight parasitic view nodes and polygons for previously probed resistances or capacitances. The parasitic prober also provides the ability to output probed parasitics to an ASCII report file and to annotate parasitic view total capacitance values to an associated schematic view.

- Opens Debugger—Investigate opens in the design.

The StarRC Cockpit is a graphical user interface that helps you to set up and execute most of the available operations. You can also use a settings file to store and load setup information.

Setting Up Virtuoso Integration

The following topics describe settings and files for correct setup of the Virtuoso Integration environment:

- Installation
- License Requirements
- Environment Variables

Installation

Two components are necessary to run StarRC Virtuoso Integration:

- The `rcskill.cxt` Virtuoso binary context file
- The StarRC base executable package

Use the following installation procedure:

1. Ensure that StarRC tool (and the IC Validator tool, if used) is contained in the operating system execution path before starting the Virtuoso tools.
2. Load the rcskill context file in the Command Interpreter Window (CIW) during Virtuoso tool invocation.

   ```
   loadContext("rcskill.cxt")
   ```

3. Initialize the context file rcskill in the Virtuoso Command Interpreter Window.

   ```
   callInitProc("rcskill")
   ```

The statements in steps 2 and 3 can be inserted into the .cdsinit file to be run automatically during startup.

Any relative path that you specify in the Cockpit must be relative to the run directory—the directory in which you invoke the Virtuoso tool. Any relative path that you specify in the RCGenParaViewBatch SKILL procedures must be relative to the StarRC run directory.

License Requirements

The Virtuoso Integration functionality requires two StarRC license keys:

- The `STAR-RC2-NETLIST` license is checked out during parasitic view generation in cdba and is not necessary for the OpenAccess flow.
- The `STAR-RC2-PROBER` license is checked out during invocation of the parasitic prober.

By default, the StarRC tool checks out a standard StarRC license. If the extraction uses a feature that requires the StarRC Ultra license, the tool checks out either an Ultra license or the appropriate number of other licenses.

If you want to use only a StarRC Custom license for the extraction, click the check box labeled “Use Custom License” in the Parasitic View Generation dialog box, as shown in Figure 50.

To use Custom licenses by default, include the following line in the .snps_settings file:

```
USE_CUSTOM_LICENSE: YES
```

Environment Variables

You can specify global operating system conditions before starting a StarRC run by setting environment variables. To set an environment variable, use the `setenv` command at

the operating system prompt with the variable name and the setting to be applied. For example:

```
% setenv RC_VI_SETTINGS_FILE starfile.txt
```

The following environment variables affect the operation of the StarRC interface with the Virtuoso application:

- `RC_VI_SETTINGS_FILE` *`file_name`*

  Specifies a Cockpit configuration file to prepopulate entry fields in the Cockpit user interface. For more information, see Populating the Cockpit Fields Automatically.

- `ADVANCED_SAVE_LOAD YES | NO`

  Specifies to modify the file load and save button configuration in the Cockpit. For more information, see Advanced Save and Load Mode.

- `RC_SAVE_ALL YES | NO`

  Specifies whether to save the Output Lib and Output Cell names to the .snps_settings file. For more information, see Output Parasitics Tab.

- `RC_ADD_MENU YES | NO`

  Specifies whether to display the StarRC pulldown menu in the Virtuoso layout and schematic windows.

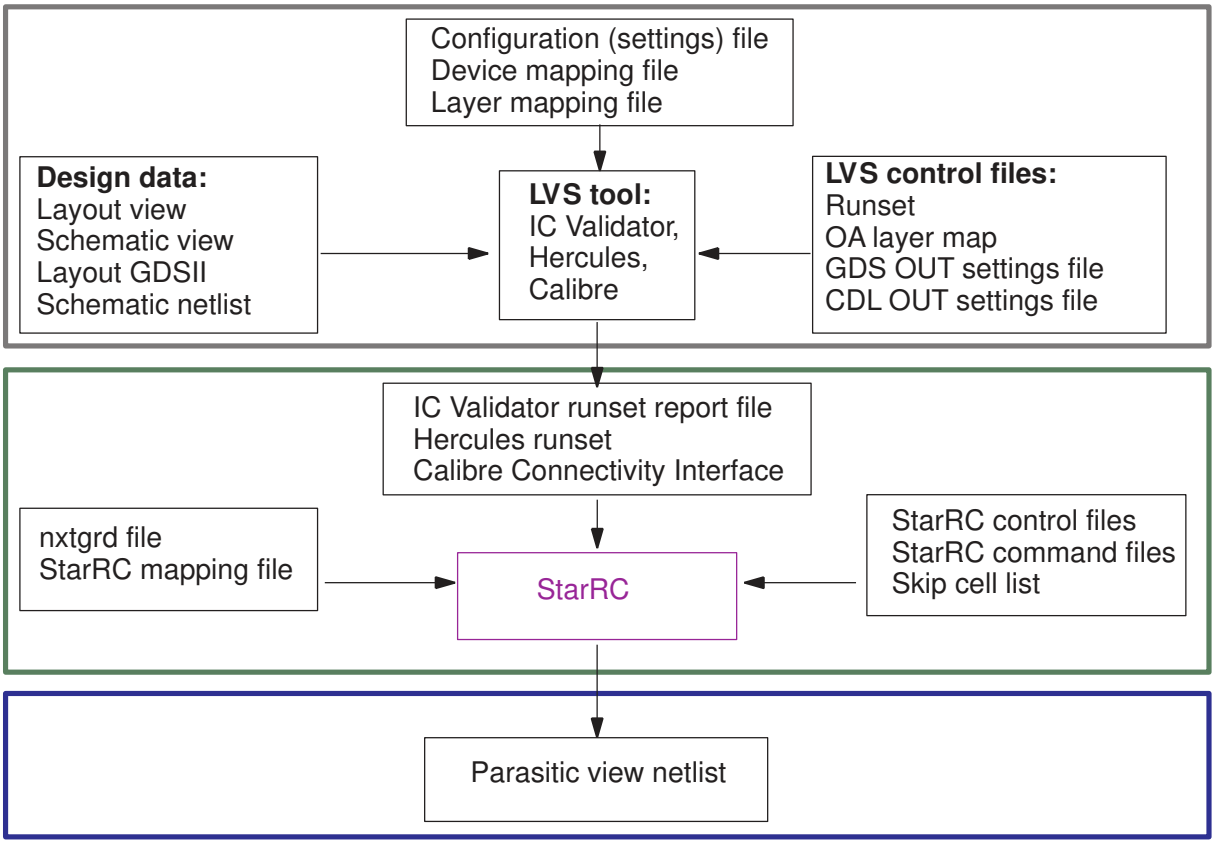
Feedback

Virtuoso Integration Flow Configuration and Related Files

The Virtuoso Integration flow has three stages, as shown in Figure 44:

- Layout versus schematic (LVS) stage, using the IC Validator, Hercules, or Calibre tools
- Extraction stage, in which you can adjust StarRC commands without a command file
- Output stage, in which you can select the output format and other options

Figure 44 Parasitic Extraction Flow in the Virtuoso Flow



The device mapping file, layer mapping file, and skip cell list are special files for Virtuoso Integration. All other files are standard input files for the LVS tools and the StarRC tool.

The Configuration or Settings File

The configuration file is a text file that contains entries for most of the settings that you can enter manually through the Cockpit graphical user interface. This file is also known as the settings file because it must have an extension of .snps_settings. Loading a settings file

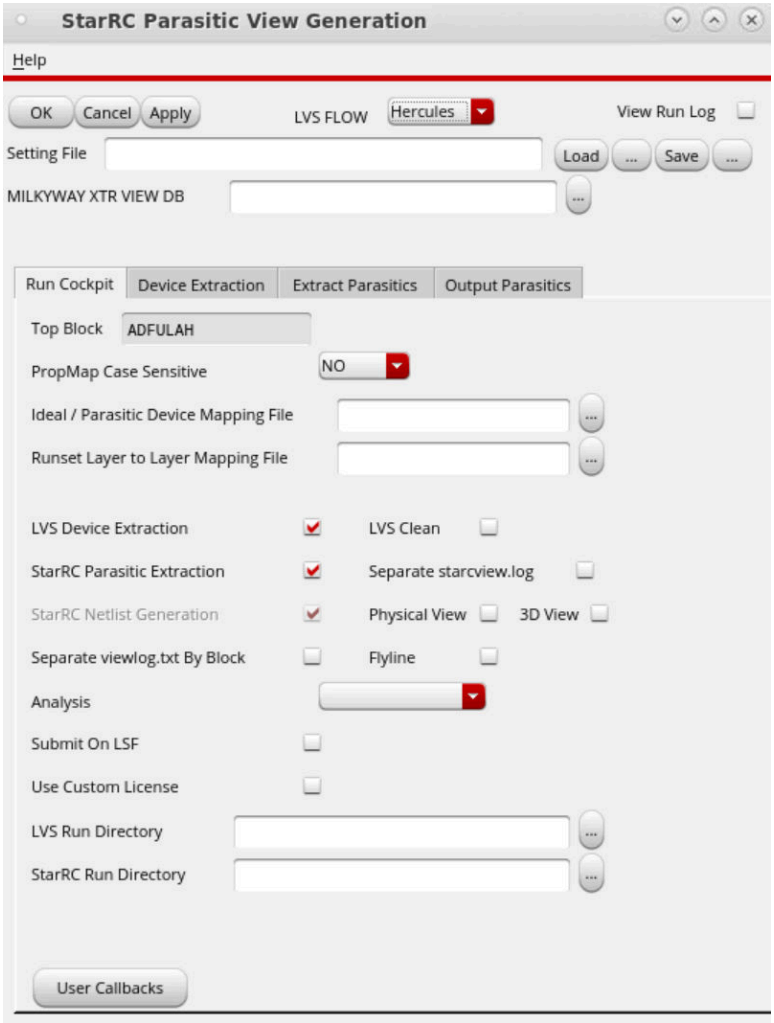
Feedback

is a convenient way to prepopulate Cockpit entry fields. In addition, saving a settings file after a session enables you to re-create the session at a later time.

Customizing the LVS Run

Choose the layout versus schematic tool (IC Validator, Hercules, or Calibre) from the Flow menu in the Parasitic Generation Cockpit, as shown in Figure 45.

Figure 45 Run Cockpit Tab



You must provide a runset for the selected LVS tool. For the Calibre Connectivity Interface flow, you must also supply a query command file. You can customize many items in the Device Extraction tab, including the following:

- IC Validator—Regardless of the setting in the runset, the IC Validator runset report file name defaults to pex_runset_report_file unless you provide a new name in the Cockpit field.
- Hercules—The location of the extract view is determined from the runset. If multiple `WRITE_EXTRACT_VIEW` commands exist in the runset, the last one is used regardless of the setting inside the runset.
- Calibre—The location of the svdb directory is determined from the runset. If multiple `MASK SVDB DIRECTORY` commands exist in the Calibre runset, the first one is used regardless of the setting inside the runset.

Customizing StarRC Extraction

To customize the extraction, you can use either of the following methods:

- Specify a StarRC command file through the StarRC Additional Commands dialog box in the Extract Parasitics tab, as shown in Figure 46.
- Use the Virtuoso Integration dialog box settings.

The priority of commands is as follows, from highest to lowest:

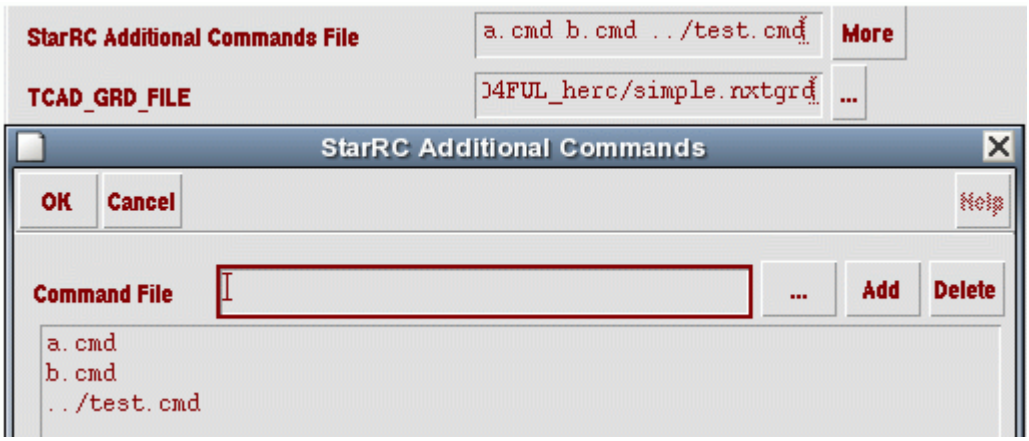
1. Required native Virtuoso Integration commands
2. Additional StarRC commands provided in a command file
3. Additional settings entered in the Virtuoso interface dialog boxes

Use the `view_cmd` or `oa_cmd` commands in the Virtuoso Integration run directory to list the StarRC commands that Virtuoso Integration has inserted into your extraction run.

For standard extraction, you must provide the nxtgrd and mapping files. However, you can run the direct ITF flow by using the `ITF_FILE` command in the .snps_settings file. In this case, the nxtgrd file specified in the user interface is ignored. For more information, see The Direct ITF Flow.

Feedback

Figure 46 Adding or Deleting Additional StarRC Command Files



Feedback

The Device Mapping File

A device mapping file maps ideal and parasitic devices in the StarRC parasitic output to the corresponding device symbols in the Virtuoso symbol libraries. This file contains an entry for every ideal and parasitic device model that exists in the parasitic output. It also provides the ability to remap standard StarRC DSPF device property names to user-specified property names. The following syntax shows the format of a single line:

```
RC_model_name dfii_lib_name dfii_cell_name dfii_view_name
     dfii_symbol_pin_1 dfii_symbol_pin_2 [dfii_symbol_pin_3] …
     [CALLBACK = procedureSymbol] [PROPMAP DSPFprop1 = ADEprop1…]
```

Argument	Definition
RC_model_name	StarRC output model name; corresponds to the schematic device model name when XREF is activated. Note that keywords pres and pcap are used for parasitic resistors and capacitors.
dfii_lib_name	Name of Virtuoso library containing the corresponding device symbol.
dfii_cell_name	Cell name of Virtuoso device symbol.
dfii_view_name	View name of Virtuoso device symbol.
dfii_symbol_pin_N	Name of the pin inside Virtuoso device symbol that corresponds to terminal #N in an analogous DSPF-based StarRC parasitic output. Note that the ordering of Virtuoso symbol pin names inside the device mapping file must match the StarRC netlist pin order for the device type of interest. The keyword `nil` can be specified for any *dfii_symbol_pin* to indicate that the terminal #N in the StarRC parasitic output should be ignored when connecting the corresponding Virtuoso library symbol.
procedureSymbol	Symbol name of an optional user-defined callback procedure that is executed before instantiating a device of type *RC_model_name*.
DSPFpropN = ADEpropN	Optional mapping of standard DSPF property name into a user-specified property name. Note that keyword PROPMAP is required before the first property name mapping entry. Setting *DSPFpropN* `= nil` prevents the listed property from being annotated to the device symbol.

Two slashes (//) serve as a comment delimiter in the device mapping file.

An example DFII symbol mapping file is as follows:

```
MNM  devlib  nmos4  ivpcell  D G S B PROPMAP l=simL w=simW
MPM  devlib  pmos4  ivpcell  D G S B PROPMAP l=simL w=simW
pres Lib  presistor  auLvs PLUS MINUS CALLBACK=insertPresProp
pcap Lib  pcapacitor auLvs PLUS MINUS CALLBACK=insertPcapProp
```

Feedback

```
pind analogLib pinductor symbol PLUS MINUS
pmind analogLib pmind symbol
```

Parasitic elements pres, pcap, pind, pmind should use the lib/cell/view from analogLib. For example:

```
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
pind analogLib pinductor symbol PLUS MINUS
pmind analogLib pmind symbol
```

However, if a parasitic element name conflicts with a user-defined device name, Virtuoso Integration provides the following parasitic element names:

- `pres[starrc]`
- `pcap[starrc]`
- `pind[starrc]`
- `pmind[starrc]`

When `pres` and `pres[starrc]` both appear in the device mapping file, `pres[starrc]` overrides `pres` as the parasitic element name.

See Also

- Instance Creation Callback

The Layer Mapping File

The layer mapping file maps layers in the StarRC mapping file to the corresponding Virtuoso technology file layers. This allows polygons, ports, and subnodes from the parasitic extraction to be stored within the generated DFII parasitic view.

Each layer that you want to store in the parasitic view should be specified in the layer mapping file and should be mapped to an existing layer-purpose pair (LPP) from the Virtuoso technology library for the library being used. The mapping file optionally lets you specify DFII layer-purpose pairs for subnode markers generated by the StarRC tool to represent parasitic top-level ports (*|P), instance ports (*|I), and subnodes (*|S).

The format of the layer mapping file is as follows:

```
RC_MAPPING_FILE_layer
   dfii_polygon_layer_name dfii_polygon_purpose_name
   [ dfii_port_layer_name dfii_port_purpose_name
   [ dfii_subnode_layer_name dfii_subnode_purpose_name]]

RC_MAPPING_FILE_layer
   dfii_polygon_layer_name dfii_polygon_purpose_name
```

```
   [ dfii_port_layer_name dfii_port_purpose_name
   [ dfii_subnode_layer_name dfii_subnode_purpose_name]]
. . .
```

Argument	Definition
RC_MAPPING_FILE_layer	Database layer name from the CONDUCTING_LAYERS, VIA_LAYERS, or MARKER_LAYERS sections of the mapping file specified by the MAPPING_FILE command.
dfii_polygon_layer_name *dfii_polygon_purpose_name*	Layer name and purpose name from the DFII technology library, which forms the layer-purpose pair to which RC_MAPPING_FILE_layer polygons should be written. If either entry is not specified or are specified as nil, polygons corresponding to the RC_MAPPING_FILE_layer is not generated within the parasitic view.
dfii_port_layer_name *dfii_port_purpose_name*	Layer name and purpose name from the DFII technology library, which forms the layer-purpose pair to which parasitic port markers corresponding to RC_MAPPING_FILE_layer interconnect should be written. Parasitic port markers are analogous to *|P nodes and .SUBCKT header nodes that would appear in the DSPF output. If either entry is not specified or are specified as nil, ports corresponding to the RC_MAPPING_FILE_layer is not generated within the parasitic view.
dfii_subnode_layer_name *dfii_subnode_purpose_name*	Layer name and purpose name from the DFII technology library, which forms the layer-purpose pair to which parasitic subnode markers corresponding to RC_MAPPING_FILE_layer should be written. Parasitic subnode markers are analogous to *|I and *|S nodes that would appear in a DSPF output. If not specified or if specified as nil, subnodes corresponding to the RC_MAPPING_FILE_layer is not generated within the parasitic view.

Use two slashes (//) to indicate comments in the layer mapping file.

Polygons are written to the parasitic view only if the IC Validator, Hercules, or Calibre database layer of the polygon is mapped to a valid Virtuoso layer-purpose pair in the DFII layer mapping file. If a layer is not listed in the mapping file, no polygons corresponding to that layer are stored in the parasitic view. If the file is not supplied at all, no graphical data is written.

Because all ports and subnodes correspond to specific database layers in standard StarRC outputs, separate layer-purpose pairs are used in the DFII layer mapping file for the generation of port and subnode markers relative to the generation of interconnect polygons. Port and subnode markers enable point-to-point resistance probing with the StarRC parasitic probing utility. Therefore, failure to include layer-purpose pairs for port and subnode markers prohibits the probing of point-to-point resistance between nodes lacking such markers.

A layer-purpose pair cannot be used both as a polygon LPP and a port and subnode LPP. If an LPP used as a polygon LPP is found in the mapping file, that LPP is disregarded if it is subsequently listed in the mapping file as a port or subnode LPP. Likewise, if an LPP used as a port or subnode LPP is found in the mapping file, that LPP is disregarded if it is subsequently listed in the mapping file as a polygon LPP.

The following example shows a DFII layer mapping file:

```
M1 metal1 net metal1 port metal1 node
fpoly poly net poly port poly node
ngate poly net poly port poly node
pgate poly net poly port poly node
diff_con cc net nil nil cc node
subtie pad drawing nil nil pad node
welltie pad drawing nil nil pad node
nsd nactive net nactive port nactive node
psd pactive net nactive port pactive node
m1_pio_marker nil nil metal1 port nil nil
m2_pio_marker nil nil metal2 port nil nil
m3_pio_marker nil nil metal3 port nil nil
poly_marker nil nil poly port nil nil
```

Parasitic View Generation

Parasitic view generation is described in the following topics:

- Net and Instance Name Conventions
- Port and Terminal Connectivity Characteristics
- Instance Property Annotation From the Schematic View
- Subnode Marker and Parasitic Device Visualization

Net and Instance Name Conventions

The StarRC parasitic view abides by naming conventions for nets and instances to enforce uniformity with DFII naming rules and naming conventions for schematic-based parasitic simulation analysis. These naming conventions are unique to the Virtuoso Integration flow and are different from standard StarRC DSPF netlist conventions.

The pipe character (|) is the default hierarchy delimiter for the parasitic view. However, the default hierarchy delimiter for the signal paths in the prelayout testbench is the slash character (/). To use the OA view along with an existing prelayout testbench in an interactive session, open the testbench and call the StarRCModHierSep() function in the command interpreter window. This function converts the slash delimiter characters to pipe characters to enable correspondence with the OA view.

Feedback

During schematic view netlist creation for LVS and StarRC schematic-based cross-referencing (using the StarRC `XREF` command), the schematic view netlist generator can append prefixes to instance names according to the conventions of the netlist generator. These prefixes, commonly called SPICE cards, propagate into the StarRC parasitic netlist when the `XREF` command is used.

However, these extra instance prefixes are not present in the original schematic view and might impede the ability to perform full matching of the parasitic view to the schematic view during simulation. Therefore, the Virtuoso Integration flow removes these instance name prefixes as follows:

- Strips the initial SPICE card from ideal (not parasitic) instance names, because the StarRC tool adds this card directly.
- For hierarchical instance names, including those preceding internal net names:
  - Decomposes the name according to the hierarchical delimiter.
  - Strips the initial X character if a decomposed instance name begins with X anywhere in the decomposed instance list.
  - If the last name in the decomposed instance (often a primitive) begins with two occurrences of the same character, strips one of them.
- If the last name in the decomposed instance (often a primitive) begins with a SPICE character such as X or M, strips that character.

Table 33 provides some examples of instance name prefix removal.

Table 33 Examples of Removal of Instance Name Prefixes

StarRC DSPF instance name	Parasitic view instance name
XI0/XI5/M1	I0/I5/1
XI0/XI5/MM1	I0/I5/1
I0/I5/M1	I0/I5/M1
I0/I5/MM1	I0/I5/M1
XI0/X15/net1	I0/I5/net1
I0/I5/net1	I0/I5/net1

Feedback

The naming convention for input data makes the following assumptions:

- The schematic netlist generator always appends an X to nonprimitive cell instances (for example, instances in the middle of a flattened hierarchical name).
- No instance name inside the schematic view begins with X.
- No instance name inside the schematic view begins with two occurrences of the same letter, such as the schematic view instance MM0.

Occurrences of bus bits (name<#>) are renamed if the bus bit is embedded within the middle of a hierarchical instance name. In such cases, the embedded string <#> is replaced with the embedded string (#). An example where this behavior can occur is an iterated schematic instance name embedded in a hierarchical name. For example, `[I0|I1|I2<3>|net4` becomes `I0|I1|I2(3)|net4`.

Port and Terminal Connectivity Characteristics

The StarRC tool reads the following information from a pre-existing view of the extracted cell and populates the same information within the parasitic view:

- netExpression parameters

  The tool parses all terminals and signals in the ports global nets view (default is layout) and records any netExpression parameters. If a terminal and a signal have the same name and both have individual netExpressions, only the netExpression of the terminal is recorded. The netExpressions are then propagated into the new parasitic view as follows:

  - If a terminal in the parasitic view has a name matching a terminal or signal name for which a netExpression was read from the existing cell, the netExpression is added to that terminal.
  - Otherwise, if a signal in the parasitic view has a name matching a cached terminal or signal name for which a netExpression was read from the existing view, the netExpression is added to that signal.

  Note:

  Terminals in the parasitic view are dictated by the presence of ports in the StarRC parasitic output which are analogous to *|P nodes or .SUBCKT headers in a DSPF netlist. If no such nodes exist, the tool does not create any terminals in the parasitic view and no netExpression parameters are transferred.

- Direction parameters for terminals

  The StarRC tool parses all terminals in the pre-existing view and records any direction parameters listed on any terminals. If a terminal in the parasitic view has the same

name as a terminal with a direction parameter in the pre-existing view, the tool attaches the same direction parameter string to the matching terminal in the parasitic view.

- isGlobal parameters for signals

  The tool parses all signals in the pre-existing view and records any isGlobal parameters listed on any signals. If a signal in the parasitic view has the same name as a signal with an isGlobal parameter in the pre-existing view, the tool attaches the same isGlobal parameter string to the matching terminal in the parasitic view. Note that the isGlobal parameter is propagated only for signals to which no terminals bearing netExpression parameters are attached. The pre-existing view from which the previous connectivity and terminal information is read is specified by the corresponding field in the StarRC parasitic generation cockpit or by the corresponding argument in the RCGenParaViewBatch procedure.

  When updating a terminal view, the StarRC tool gathers terminal information from a given symbolic view and parses all signals in the parasitic view to check for floating nets on device instance terminals. If a net name matches one terminal name on the symbol view, the tool creates the name as the terminal name for the parasitic view.

- Power net name for the ports

  The StarRC tool creates the ports of the parasitic view as the input database top-block port names.

  When integrating the parasitic view into another circuit, some additional power pins might be necessary for the parasitic view to connect to upper-level views. If additional power port names are necessary for the parasitic view, use the “Reading Pin from symbol” option on the Cockpit to assign an additional (symbol) view for the StarRC tool to extract the additional power net names from the ports of the given view. You can then create new ports with the name for the parasitic view.

Instance Property Annotation From the Schematic View

Properties on the instance inside the parasitic views come from one of two places, either the schematic view or the LVS tool. There might also be custom usages, such as to copy a property value to multiple property names, delete (make nil) a property, change a property name, or create a new property. The StarRC tool needs to set the property names and values in the parasitic view for the simulator to work correctly.

Controlling Instance Property Annotation

Use the `XREF:YES` command to generate a parasitic view constructed by the layout view and the extracted netlist from the LVS tool, with the schematic net name, instance name, or instance property attached.

The following options control the default behavior of instance property annotation:

- In the .snps_settings file

  ```
  CARRY_SCH_PROPERTY : [YES]|NO
  ```

- In the RCGenParaViewBatch procedure

  ```
  carry_sch_property : [t]/nil/noPcell
  ```

If both the settings file and the batch procedure are specified, the batch procedure takes precedence. Table 34 describes the effects of the settings.

Table 34 Settings For Instance Property Annotation

Settings file (CARRY_SCH_PROPERTY)	Batch procedure (carry_sch_property)	Result
`YES`	`t`	Schematic properties are carried to the OA view for LVS devices. PCells are individually evaluated.
`NO`	`nil`	No schematic properties are carried.
not applicable	`noPcell`	PCells in the design are skipped. Schematic properties are carried from the master instance.

You might encounter the following issues:

- Some properties only exist in the schematic view.

  Because LVS tools do not take schematic views as input, they use the CDL netlist generated from the schematic view. Some properties might not be included in the CDL netlist procedure. Also, LVS tools report only the properties that have been defined in the LVS runset to LVS results database. Therefore, the StarRC tool must carry those properties and their values from the schematic to the parasitic view.

- Some property names are used in both the schematic view and LVS results.

  In this case, the layout view property values override the schematic view property values.

For example, consider a schematic view that contains the following two instances:

```
I1: p1=s1 p2=s1 p3=s1 p4=s1
I2: p1=s2 p2=s2 p3=s2 p4=s2
```

The LVS tool extracts the following information:

```
I1: p3=l1 p4=l1 p5=l1
I2: p3=l2 p4=l2 p5=l2
```

In the DFII library, cells I1 and I2 are used the cell m.

```
m: p1=0 p2=0 p3=0 p4=0
```

When `CARRY_SCH_PROPERTY:YES` is specified, the parasitic view contains the following:

```
I1: p1=s1 p2=s1 p3=l1 p4=l1 p5=l1
I2: p1=s2 p2=s2 p3=l2 p4=l2 p5=l2
```

When `CARRY_SCH_PROPERTY:NO` is specified, the parasitic view contains the following:

```
I1: p1=0 p2=0 p3=l1 p4=l1 p5=l1
I2: p1=0 p2=0 p3=l2 p4=l2 p5=l2
```

Property Mapping

This section describes syntax to adjust property annotation behaviors. These behaviors are applied to all properties, in both the schematic view and the LVS results.

Table 35 Settings for Property Mapping

Syntax	Description
`dspfProp`	Property name or value in the StarRC DSPF results, usually from LVS results
`schProp`	Property name or value In the schematic view
`cdfProp`	Property name defined in the CDF file
`preProp`	Property name or value before StarRC parasitic view generation
`paraProp`	Property name or value in the final parasitic view

Property Mapping Behavior

The following examples describe property mapping operations. This information is for reference only; use the file provided by the foundry.

A=B

In the parasitic view, paraProp B gets its value from preProp A. The paraProp A value also comes from preProp A. This is equivalent to A=A and A=B.

A=B and A=C

Similar to A=B; for multiple specifications.

A>B

Two actions occur: paraProp B gets its value from preProp A and paraProp A is removed. If A is a cdfProp and cannot be removed, the value is set to nil such that A=nil and A=B.

A="*constant*"

A constant is assigned to paraProp A, regardless of the original value. If there is no paraProp with the name A, the tool creates one with the assigned value.

A=nil

Remove paraProp A; if A is a cdfProp and cannot be removed, the value of A is set to nil.

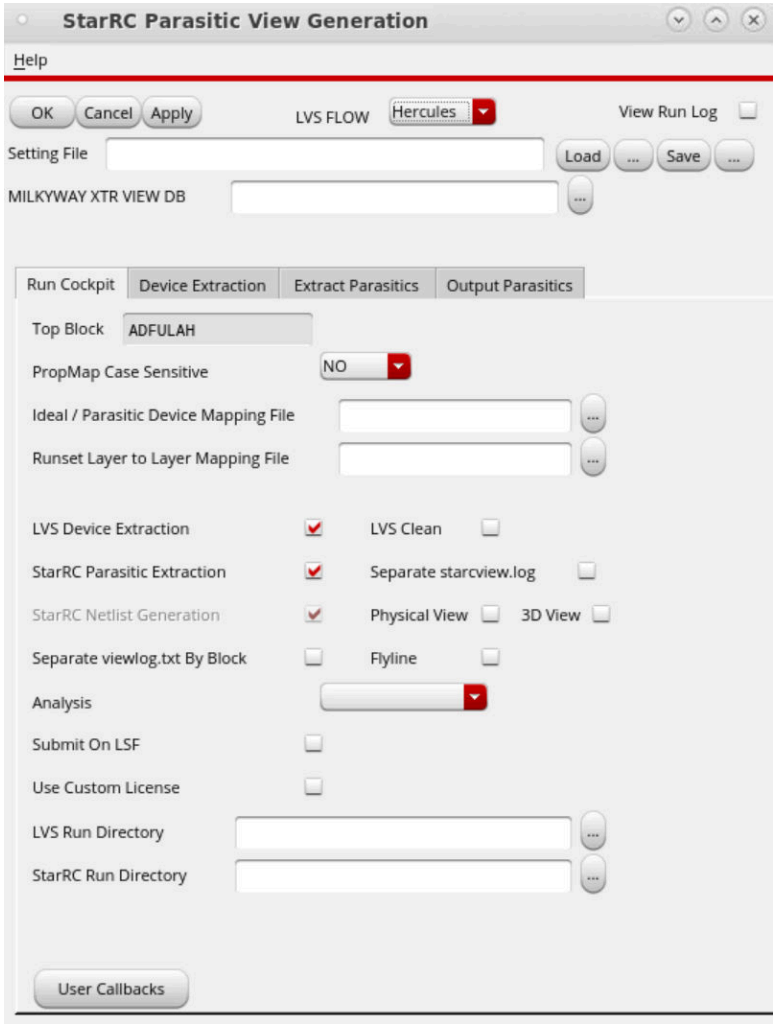
Feedback

PROPMAP Case Sensitivity

You can specify the case sensitivity in the following ways:

- In the Cockpit dialog box with the PropMap Case Sensitive option, as shown in Figure 47.

Figure 47 Run Cockpit Tab



- In the .snps_settings file:

```
PROPMAP_CASE_SENSITIVE : YES | NO | MIXED
```

- In the RCGenParaViewBatch procedure:

```
?propmap_case_sensitive t | [nil] | mixed
```

Feedback

A setting of `YES` or `t` means that all property names are kept unchanged. A setting of `NO` or `nil` means that all property names are converted to lowercase. A setting of MIXED means that property names are kept unchanged except for the following property names, which are converted to lowercase: l, w, as, ad, ps, pd, nrd, nrs, m, area, and pj.

The settings are equivalent to the settings of the StarRC `OA_PROPMAP_CASE_SENSITIVE` command, as shown in Table 36. If multiple settings exist, the SKILL command takes precedence.

Table 36 SKILL and StarRC Command Settings

?propmap_case_sensitive setting	OA_PROPMAP_CASE_SENSITIVE setting
t	YES
nil	NO
mixed	MIXED

This option controls the matching behavior of `PROPMAP` commands.

For example,

```
PROPMAP ABC=abc
```

When `PROPMAP_CASE_SENSITIVE : YES`

If there are 2 `preProp` named as ABC and abc, only ABC is copied to abc.

```
preProp : ABC=10 abc=100
paraProp: ABC=10 abc=10
```

When `PROPMAP_CASE_SENSITIVE : NO`

```
preProp: Abc=10
```

StarRC uses the Abc value as ABC to write to abc.

```
paraProp: Abc=10 abc=10
```

Instance Name Matching Rule

The StarRC tool performs instance property annotation by finding the corresponding instance in the schematic view. The tool also performs SPICE card stripping of schematic view instances, then finds the corresponding instance in the schematic view to obtain the properties for annotation.

When the tool cannot find a schematic view instance to annotate properties to parasitic view instances, the failed instances are reported in a file named schematic_info_log.

Feedback

Subnode Marker and Parasitic Device Visualization

The runset layer to DFII layer mapping file provides the ability to write extracted polygon data to the parasitic view. Besides the storage of interconnect polygons, graphical data including subnode markers, port markers, and pres or pcap flylines are also written to the parasitic view. For more information about the layer mapping file, see The Layer Mapping File.

A subnode is defined as any *|I or *|S node that would normally occur in a standard StarRC DSPF parasitic netlist. A port is analogous to any *|P node or any entry in the .SUBCKT header line of a standard DSPF parasitic netlist. These nodes represent the electrical connection points for parasitic devices and ideal devices. Every subnode has unique xy coordinates as well as an extracted database layer on which the subnode lies. Using this information, it is possible to represent the subnodes in the parasitic view with small marker shapes placed at their corresponding xy coordinates.

The DFII layer-purpose pair to which a port or subnode marker is written is defined in the DFII layer mapping file. Only ports or subnodes whose corresponding database layers are listed in the DFII layer mapping file has markers written to the parasitic view. The default size of all subnode markers is 0.1 um x 0.1 um. This default size can be changed by adding an entry to the Cockpit configuration file as follows:

```
SUBNODE_SIZE: subnode_side_length_in_microns
```

Graphical data is also stored for parasitic resistors and coupling capacitors in the form of flylines between subnodes. Flylines are not stored for grounded capacitors because such capacitors by definition do not terminate at a finite xy coordinate on an interconnect polygon. These flylines are annotated with two properties: the parasitic instance name and the parasitic value. All flylines for parasitic resistors are written to a single Virtuoso layer-purpose pair. Likewise, all flylines for parasitic capacitors are written to a separate Cadence layer-purpose pair. These layer-purpose pairs can be listed in the DFII layer mapping file as follows, using the special tags *pres and *pcap.

```
// <*pres or *pcap> dfii_polygon_lyr dfii_polygon_purpose
*pres poly net
*pcap metal1 net
```

If no *pres and *pcap lines are defined in the DFII layer mapping file, the StarRC tool uses the following default mapping:

```
*pres y0 drawing
*pcap y1 drawing
```

Note:

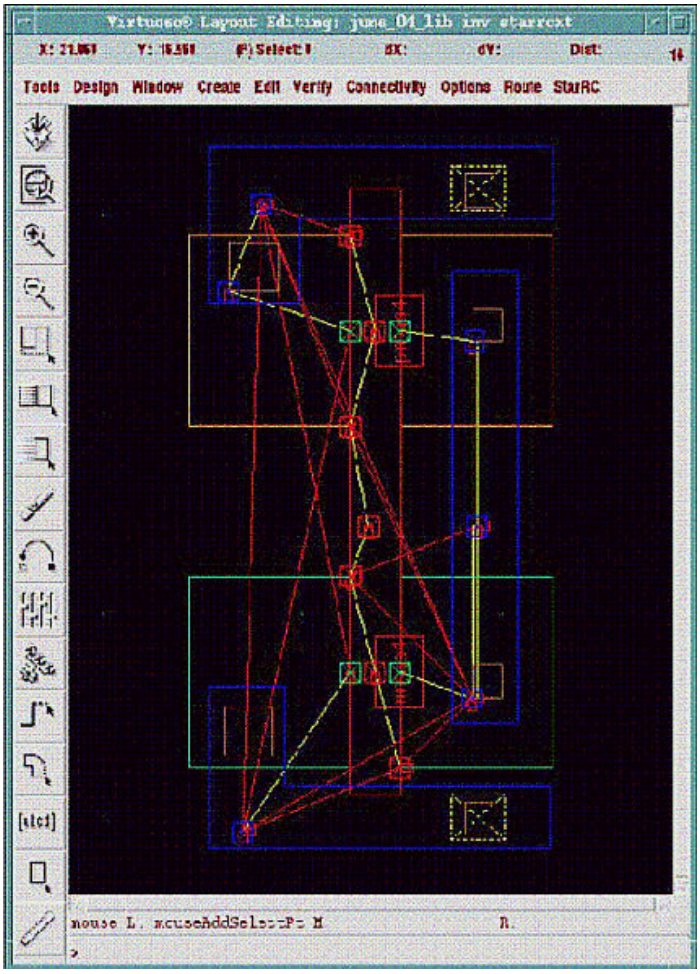
A flyline is stored in the parasitic view only if both of its nodes have markers stored in the parasitic view. Likewise, port and subnode markers are only stored for runset layers that are mapped in the DFII layer mapping file. Therefore, the

Feedback

number of parasitic resistor and capacitor flylines present in the parasitic view is a direct function of the runset layer to DFII layer mappings in the DFII layer mapping file.

An example of the StarRC parasitic view containing subnode markers and flylines is shown in Figure 48.

Figure 48 StarRC Parasitic View With Port and Subnode Markers and Pres or Pcap Flylines



Feedback

User-Defined Callbacks

You can customize StarRC parasitic view generation through the use of user-defined callbacks. A callback is a method to invoke a procedure or command at a specific event in the flow. Four types of callbacks are available and are covered in the following sections:

Virtuoso tool versions 6.x and later uses an Open Access (OA) file writer to generate parasitic views. This flow supports the pre-extraction and postprocessing callbacks but does not support preprocessing or instance creation callbacks. For Virtuoso versions 6.x and later, you can disable the OA file writer and use the SKILL writer instead, which enables the use of all callback types. To disable the OA file writer, use one of these methods:

- Edit the .snps_settings file to include the following line:

  ```
  OA_WRITER:NO
  ```

- Edit the RCGenParaViewBatch procedure to include the following line:

  ```
  oa_writer nil
  ```

Pre-LVS Callback

The pre-LVS callback is a SKILL expression that is loaded and executed before the layout versus schematic (LVS) step. This callback interface enables you to perform customized operations on StarRC inputs before performing LVS. You can specify a callback command string in the appropriate field inside the StarRC Cockpit. Alternatively, the command string can be automatically loaded from the cockpit configuration file by including the following statement:

```
PRE_LVS_CALLBACK: cmd_string
```

The *cmd_string* argument can be any type of procedure call or SKILL expression. You can configure the pre-LVS callback using the following predefined symbols:

Symbol	Definition
lvs_rundir	LVS Run Directory
starrc_rundir	StarRC Extraction Directory
cci_runset	Calibre runset file for LVS
cci_query_file	Calibre query file for Calibre Connectivity Interface database
herc_runset	Hercules runset for LVS

The following example executes a call to procedure UserPreLvsCB, which uses the lvs_rundir symbol as an argument.

```
PRE_LVS_CALLBACK: UserPreLvsCB(lvs_rundir)
```

Pre-Extraction Callback

The pre-extraction callback is a SKILL expression that is loaded and executed before the beginning of StarRC view generation. This callback interface enables you to perform customized operations on StarRC inputs before beginning the extraction. If the LVS and StarRC steps are used consecutively, pre-extraction callback is executed after LVS finishes and before extraction starts. You can specify a callback command string in the appropriate field inside the StarRC Cockpit. Alternatively, the command string can be automatically loaded from the cockpit configuration file by including the following statement:

```
PRE_EXTRACTION_CALLBACK: cmd_string
```

The *cmd_string* argument can be any type of procedure call or SKILL expression. You can configure the pre-extraction callback to use existing LVS results by using the following predefined symbols:

Symbol	Definition
lvs_rundir	LVS Run Directory
starrc_rundir	StarRC Extraction Directory
cci_runset	Calibre runset file for LVS
cci_query_file	Calibre query file for Calibre Connectivity Interface database

Feedback

Symbol	Definition
herc_runset	Hercules runset for LVS

The following example executes a call to procedure UserPreExtractionCB, which uses the lvs_rundir and cci_query_file symbols as arguments. These symbols are only valid if the LVS process is already included in the tasks.

```
PRE_EXTRACTION_CALLBACK: UserPreExtractionCB(lvs_rundir cci_query_file)
```

View Preprocessing Callback

The view preprocessing callback is a SKILL expression that is executed after StarRC extraction and before parasitic view generation. You can specify a callback command string in the appropriate field inside the StarRC Cockpit. Alternatively, the command string can be automatically loaded from the cockpit configuration file by including the following statement:

```
PREPROCESS_CALLBACK: cmd_string
```

The *cmd_string* argument can be any type of procedure call or SKILL expression. The following symbols can be used in the argument string as variables or procedure arguments:

Symbol	Definition
cellview	A dbObject of new empty parasitic cell view before it is populated with parasitics and physical shapes
cmdfile	String object representing the name of the StarRC command file
usersym	Generic symbol which you can set and then evaluate in downstream callback code

The following example executes a call to procedure UserPreProcCallback, which uses the cellview and cmdfile symbols as arguments:

```
PREPROCESS_CALLBACK: UserPreProcCallback( cellview cmdfile )
```

View Postprocessing Callback

The view postprocessing callback is a SKILL expression that is executed after the parasitic cell view is populated with parasitics and shapes but before it is saved and closed.

You can specify a callback command string in the appropriate field inside the StarRC Cockpit. Alternatively, the command string can be automatically loaded from the cockpit configuration file by including the following statement:

```
POSTPROCESS_CALLBACK: cmd_string
```

The *cmd_string* parameter can be any type of procedure call or SKILL expression. The following symbols can be used in the argument string as variables or procedure arguments:

Symbol	Definition
cellview	A dbObject of the parasitic cell view after it is populated with parasitics and physical shapes
cmdfile	String object representing the name of the StarRC command file
usersym	Generic symbol that is set by you in upstream code and then evaluated inside the postprocessing callback

The following example executes a call to procedure `UserPostProcCallback`, which uses the `cellview` and `usersym` symbols as arguments:

```
POSTPROCESS_CALLBACK: UserPostProcCallback( cellview usersym )
```

When you use the *cellview* symbol, the callback applies to all cell views that were created by the StarRC tool, including all of the corners of a simultaneous multicorner extraction.

Instance Creation Callback

You can use an instance creation callback procedure to manipulate instance parameter lists, names, coordinate locations, and orientations before placing the instance. A procedure name can be specified on a model-by-model basis in the DFII_DEVICE_MAP file using the optional parameter CALLBACK=*procedureSymbol*. This procedure is invoked before creating an instance of the corresponding model type.

The user-defined procedure identified by *procedureSymbol* must be defined to accept two arguments as follows:

Argument	Definition
argument_1	A defstruct instance that points to a property list of all properties specific to the instance.

Feedback

Argument	Definition
argument_2	A generic symbol that you can manipulate within the view-level preprocessing/postprocessing procedures and then call within the instance-level procedures; corresponds to the symbol usersym within the code scope in which the preprocessing/postprocessing/instance-level procedures are invoked.

The defstruct property list for argument_1 is as follows:

argument_1 property	Definition
inst	Instance name of device; type = string
coordlist	XY coordinates of the instance; format is list `(xcoord ycoord)`
orientation	Orientation of instance (for example, R0, R90); type = string

propList	List of parameters that are being annotated to the instance; format is
	`list(list(t_propname1 t_propType1 g_value1)`
	`list(t_propName2 t_propType2 g_value2) …)`

instNodes	Read-only list of terminal node names; type=list. An example is: `list ("M1|DRN" "M1|GATE" "M1|DRN" "M1|BULK")`

Callback Flow Example

The following example shows a user-defined callback procedure to instantiate a new user-defined property on each MPM device type, depending on a setting in the StarRC command file.

```
; set via_cap property on disembodied property list if
; EXTRACT_VIA_CAPS was used in the StarRC run
procedure( UserParseCmdFile( cmdfile usersym )
 let( ( str stream fields )
   stream = infile( cmdfile )
   while( gets( str stream )
      fields = parseString(str,": \n")
      if( nth(0 fields) == "EXTRACT_VIA_CAPS" &&
          nth(1 fields) == "YES"

         then putprop( usersym t 'via_cap )
      )
      )
   )
)

procedure( UserAddEVCProp( dev usersym )
   if( usersym->via_cap
```

```
      then dev->propList =
         cons( list("viacap" "string" "TRUE" ) dev->propList )
   )
```

Specify the instance creation callback within DFII_DEVICE_MAP as follows:

```
MPM devlib pmos4 ivpcell D G S B CALLBACK=UserAddEVCProp
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
```

Specify a preprocessing callback procedure call in the cockpit configuration file as follows:

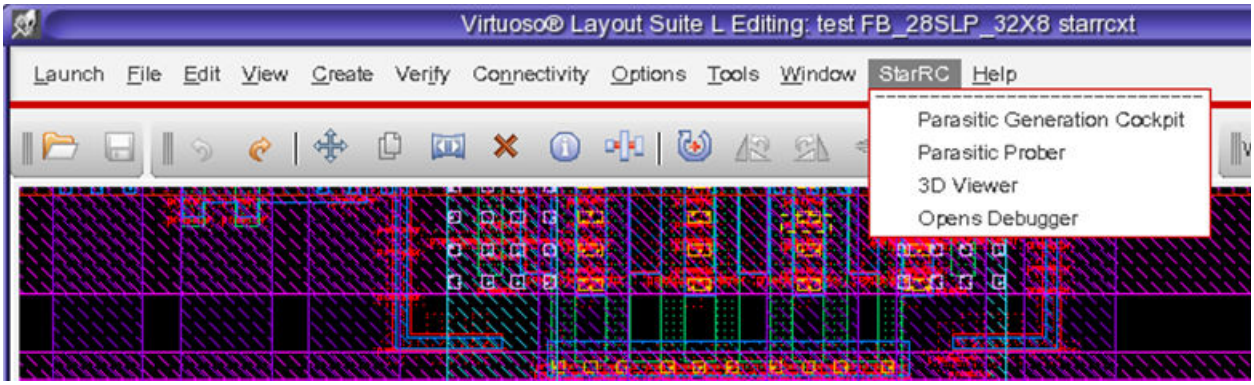
```
PREPROCESS_CALLBACK: UserParseCmdFile( cmdfile usersym )
```

The result of this setup is that devices of type MPM has a string property called viacap in the parasitic view if `EXTRACT_VIA_CAPS: YES` was set in the StarRC run.

StarRC Parasitic Generation Cockpit GUI

Select the StarRC Parasitic Generation Cockpit by choosing StarRC > Parasitic Generation Cockpit from the Virtuoso menu bar, as shown in Figure 49.

Figure 49 Starting the StarRC Parasitic Generation Cockpit in Virtuoso



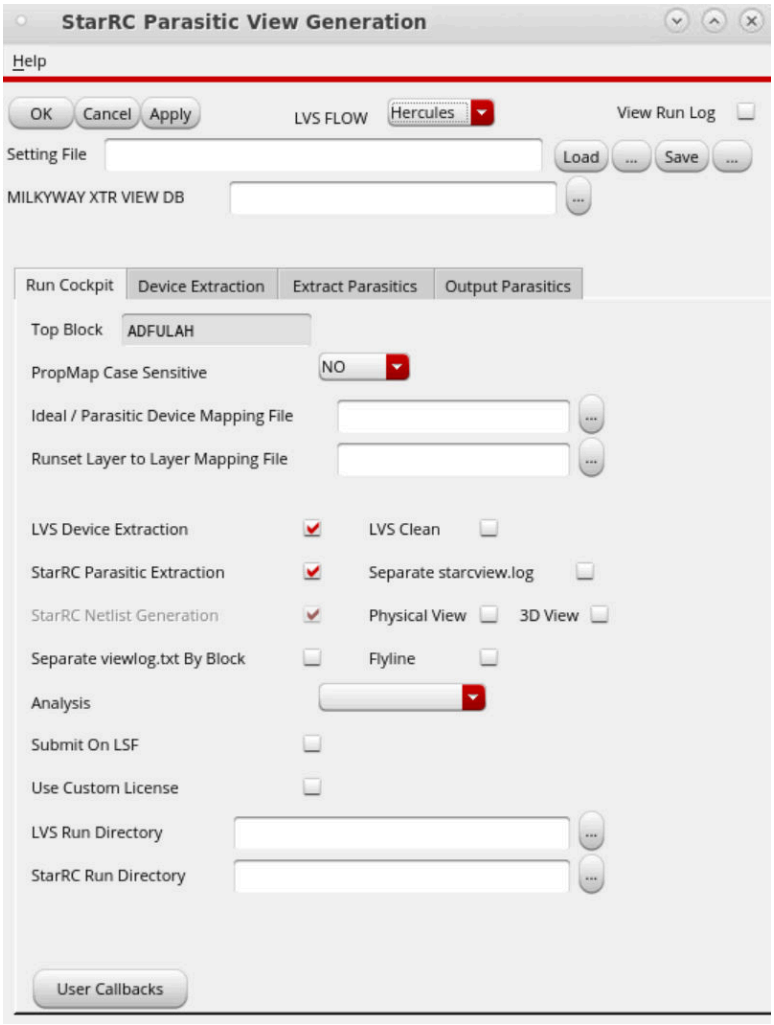
Each tab in the Cockpit represents a step in the flow; click a tab to see the options relevant for a specific step.

Figure 50 shows the StarRC Parasitic View Generation dialog box, also called the Cockpit. From the Cockpit, you can execute the LVS to extraction flow either as a complete unit or

incrementally in separate stages. You can also regenerate netlists or parasitic views if an extraction run has already been performed.

To see the real-time results of the LVS tool or StarRC run, check the View Run Log check box.

Figure 50 Run Cockpit Tab



Feedback

The top portion of the window is slightly different for different LVS tools, as shown in Figure 51.

Figure 51 Interface Options For the IC Validator and Calibre LVS Tools

Some fields require file names or directory names. Use the button labeled "…" to open a file browser instead of manually entering the names.

One of the selections on the Run Cockpit tab is a checkbox to enable the GPD flow, as shown in Figure 50. The GPD is a binary database that offers faster runtime and smaller disk space compared to generating conventional netlists.

You can also enable the GPD flow as follows:

- In the .snps_settings file

```
ENABLE_GPD_FLOW : YES|[NO]
```

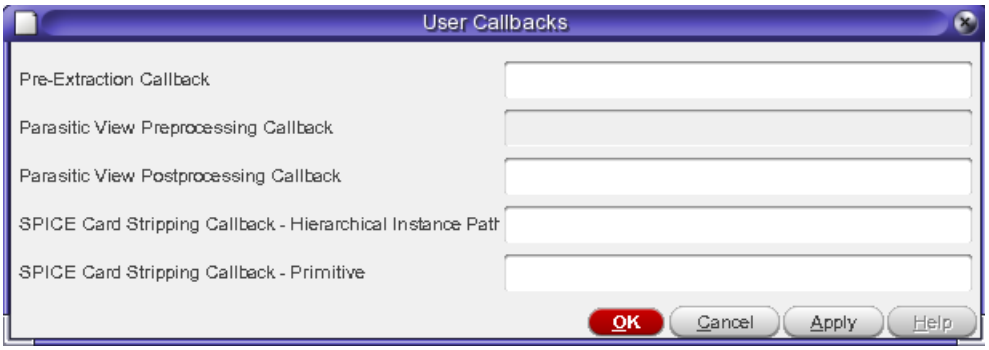
- In the RCGenParaViewBatch procedure

```
enable_gpd_flow : t/[nil]
```

The User Callbacks button, located near the bottom of the Run Cockpit tab, allows you to specify the names of callback procedures, as shown in Figure 52. For more information about callback procedures, see User-Defined Callbacks.

Feedback

Figure 52 User Callback Selection



Populating the Cockpit Fields Automatically

You can use a cockpit configuration file to automatically populate many of the fields in the Cockpit. The configuration file is also known as a settings file and must have an extension of .snps_settings. Specify a settings file as follows:

- Set the `RC_VI_SETTINGS_FILE` environment variable to point to the configuration file.
- If the `RC_VI_SETTINGS_FILE` variable is not set, the Cockpit looks for the .snps_settings file in the directory from which Virtuoso was invoked.

Choose Load or Save to open a file browser and select a setting file. You can also type the file name in the Setting File box.

Use one of the following formats for each entry in the configuration file:

```
option_1 : value_1
CONSTANT option_2 : value_2
```

The `CONSTANT` keyword makes the entry for that option not editable in the user interface and can be inserted at the beginning of any line in the settings file.

Use a semicolon (;) to indicate the beginning of a comment inside the .snps_settings file. In the following example, the second line is ignored:

```
CONSTANT TCAD_GRD_FILE: simple.nxtgrd
; CONSTANT TCAD_GRD_FILE: simple2.nxtgrd
```

In this example, the `-hier` and `-spice` options are ignored:

```
CALIBRE_LVS_CMD_LINE_OPT : -lvs ; -hier -spice
```

Table 37 lists the Cockpit field options and values that can be automatically populated.

Feedback

Table 37 Cockpit Fields and Options That Can Be Prepopulated

Tab name	*field_option : field_value*
Run Cockpit	DFII_DEVICE_MAP: *filepath* DFII_LAYER_MAP: *filepath* FLOW: ICV | HERCULES | CALIBRE PREPROCESS_CALLBACK: *SKILL_expression* POSTPROCESS_CALLBACK: *SKILL_expression*
Device Extraction (IC Validator flow)	ICV_RUNSET: *filepath* ICV_RUNSET_REPORT_FILE: *filepath*
Device Extraction (Hercules flow)	HERCULES_RUNSET: *filepath* HERCULES_COMMAND_LINE_OPTIONS: *command_string*
Extract Parasitics	TCAD_GRD_FILE: *filepath* MAPPING_FILE: *filepath* CALIBRE_QUERY_FILE: *filepath* (Calibre flow) CALIBRE_RUNSET: *filepath* (Calibre flow) EXTRACTION: R | C | RCCOUPLE_TO_GROUND: YES | NO NETLIST_GROUND_NODE_NAME: *string*
Additional Options	Any StarRC command file option listed in the Additional Options dialog box
Other	SUBNODE_SIZE: *subnode_side_length_in_microns* PORT_ANNOTATION: yes | no PORT_ANNOTATION_VIEW: *lib_cel_view*

By default, the StarRC commands specified in the Cockpit, such as the Additional Option dialog box, are saved to the .snps_settings file. The GUI settings are also saved to the .snps_settings file. To create a new .snps_settings file, use the following procedure:

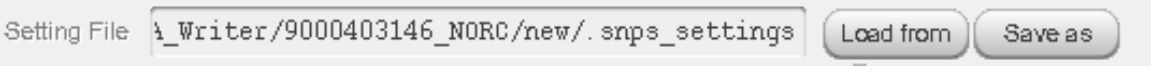
1. Open a blank Cockpit dialog box.
2. Edit the fields in the Cockpit.
3. Execute the run.
4. If the run is successful, save the settings to a new .snps_settings file. The new .snps_settings file contains all required settings to reproduce a run.

Feedback

Advanced Save and Load Mode

The interface for saving or loading a settings file is shown in Figure 53.

Figure 53 Advanced Save and Load Mode



To enable this feature, set the ADVANCED_SAVE_LOAD environment variable to YES:

```
$ setenv ADVANCED_SAVE_LOAD YES
```

Functions in the StarRC Parasitic View Generation Dialog Box

Table 38 describes the commands and options in the top section of the StarRC Parasitic View Generation Dialog Box.

Table 38 Commands and Options for StarRC Parasitic View Generation

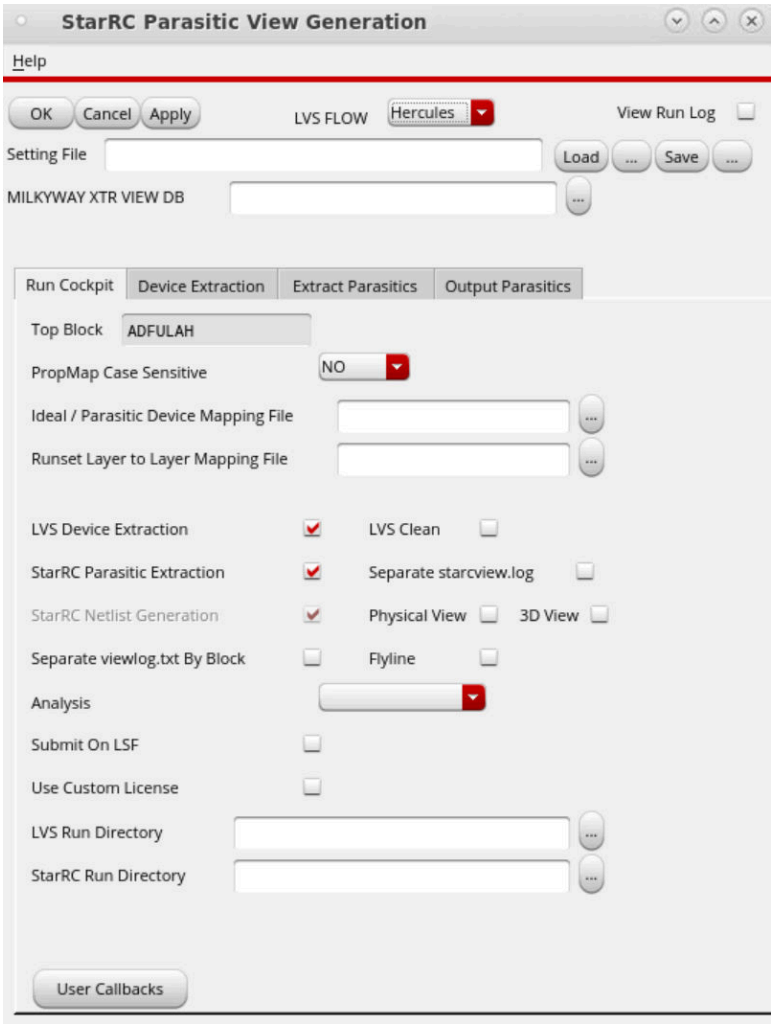
Command or option	Description
OK	Starts a Cockpit job and close Cockpit window
Cancel	Closes the Cockpit window
Apply	Starts Cockpit job and keep the dialog box open
Flow	Specifies one of three LVS tools and changes the options in the Device Extraction Tab accordingly for the specified flow
Setting File	Points to a .snps_settings file
Select	Opens the file browser to display a setting file
Load	Populates the Cockpit fields with values specified by the setting file
Save	Saves the current Cockpit values to the file in the Setting Files box
Milkyway XTR VIEW DB ICV RUNSET REPORT FILE CALIBRE_RUNSET CALIBRE_QUERY_FILE	Specifies an LVS result database

Feedback

Run Cockpit Tab

An example of the Run Cockpit tab is shown in Figure 54.

Figure 54 Run Cockpit Tab



Some of the settings on this tab are as follows:

LVS Clean

If you select LVS Clean, Virtuoso Integration Cockpit execution checks if the LVS job is compared or not. If not, Virtuoso Integration stops the job. The StarRC tool obtains the LVS results from the following files:

- topblock.LVS_ERRORS (in the IC Validator and Hercules flows)
- svdb database (in the Calibre flow)

Physical View

The parasitic view consists of two parts, the physical view and the logical view. Use the physical view for browsing and probing; use the logical view for simulation and schematic view probing.

The checkbox on the Run Cockpit tab controls the physical view generation. If it is not selected, no physical view is generated, which saves runtime.

After a physical view is generated, you can access it from the Create menu in the top-level Virtuoso menu.

Flyline

A flyline is a line that connects nodes on the same net. It helps you to probe point-to-point resistance. Generating a flyline and storing it in the parasitic view consumes runtime and disk space. By default, flyline generation is disabled.

LVS Run Directory

The directory in which to execute the LVS run. All output files are written to the same directory.

StarRC Run Directory

The directory in which to execute a StarRC extraction run.

User Callbacks

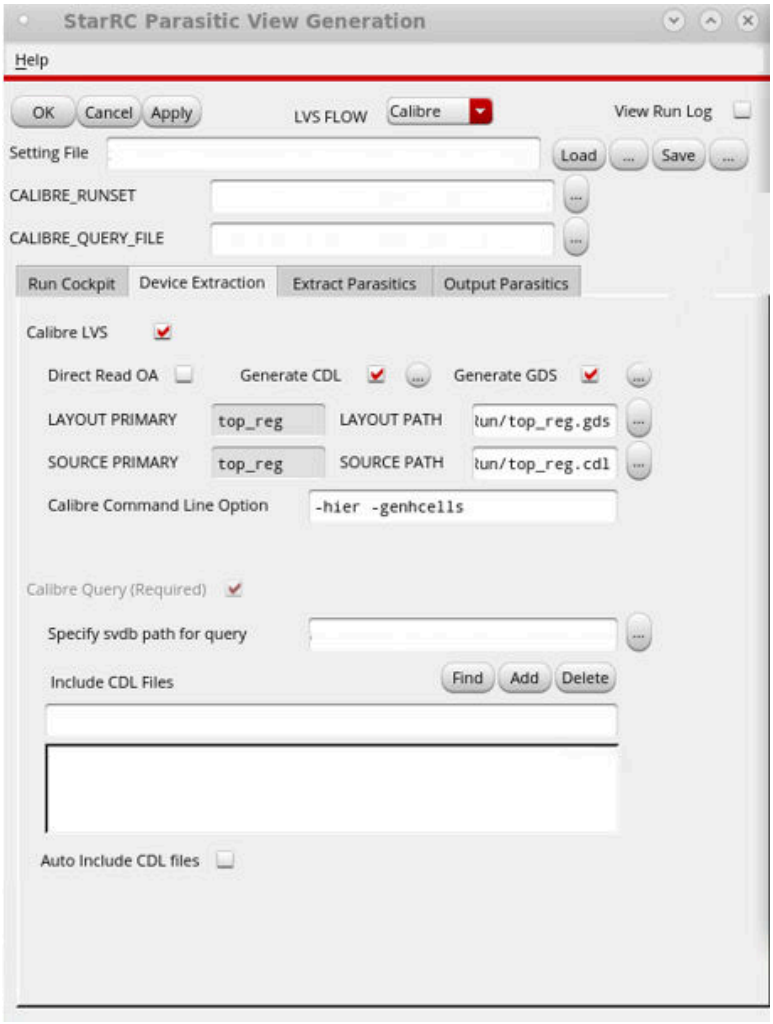
Five kinds of callbacks are available. For more information, see User-Defined Callbacks.

Feedback

Device Extraction Tab

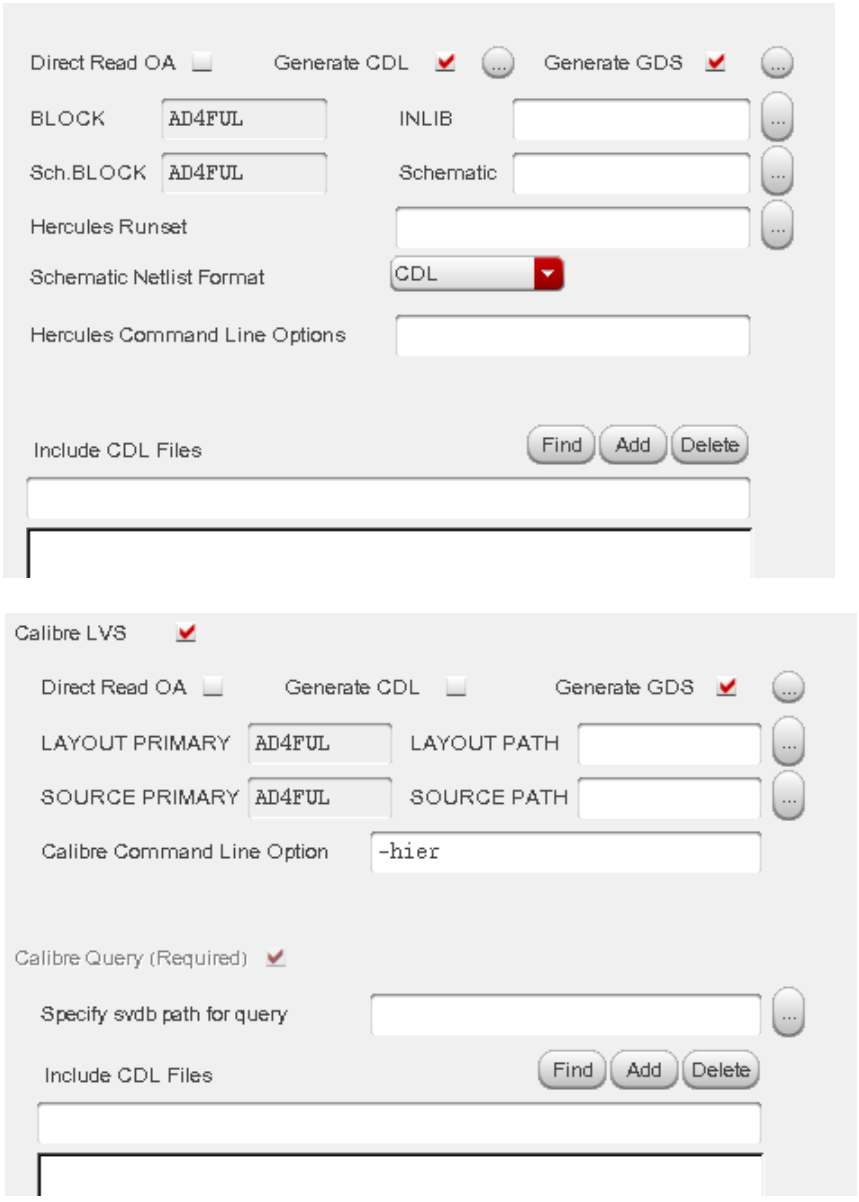
The Device Extraction tab is shown in Figure 55 for the IC Validator flow.

Figure 55 Device Extraction Tab for IC Validator Flow



The Device Extraction tab appearance is slightly different for other LVS tools, as shown in Figure 56.

Figure 56 Device Extraction Tab for Hercules and Calibre Flows



Direct Read OA

When you select this option, the LVS tool directly reads the OA layout view. Other options allow you to choose a view other than layout view or to add a layer mapping file. For information about the mapping file usage and syntax, see the documentation for the LVS tool.

Generate CDL

When you select this option, the netlist is written to a CDL file that is passed to the LVS tool, instead of to a runset- or Cockpit-specified netlist file.

You can modify the streaming option in the CDL out settings dialog box, as shown in Figure 57.

Generate GDS

When you select this option, Virtuoso Integration streams out the layout view to a GDSII file that is passed to the LVS tool, instead of streaming the layout to the runset- or Cockpit-specified GDSII file.

You can modify the streaming option in the GDSII stream out settings dialog box, as shown in Figure 58.

Include CDL Files

To use multiple CDL files as input to the Virtuoso Integration LVS tools, specify the CDL files in the GUI.

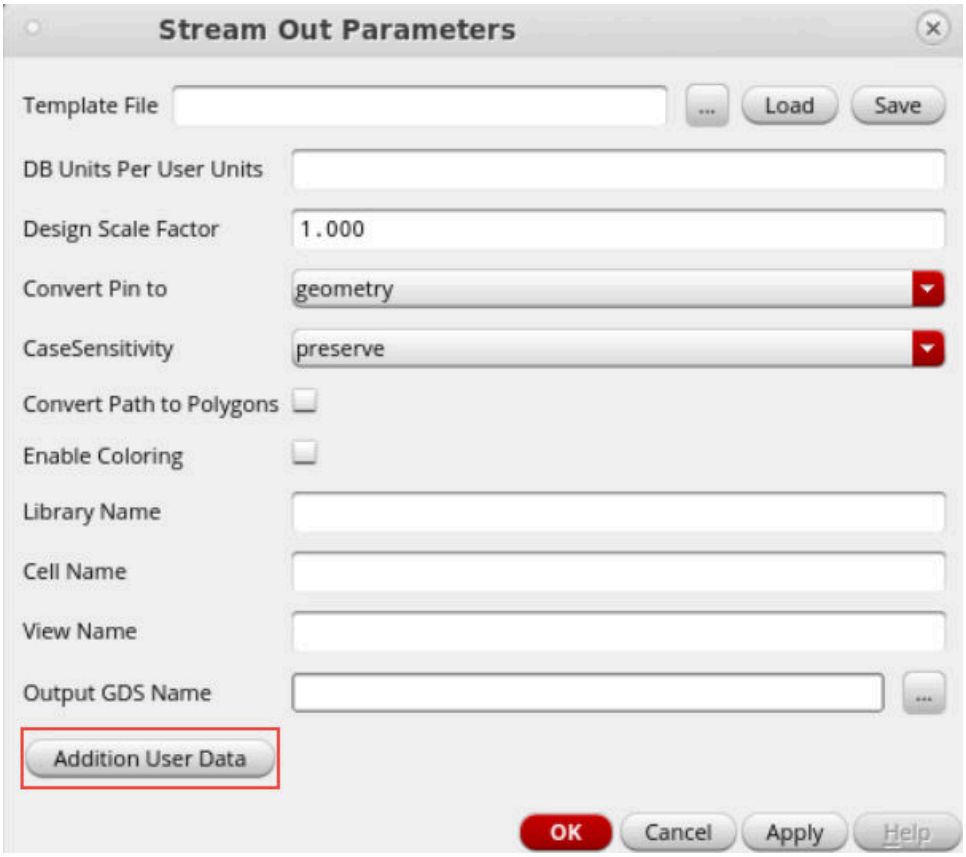
The View Run Log check box near the top of the Device Extraction tab displays the StarRC log file in a separate window.

Feedback

Figure 57 CDL Out Settings

Feedback

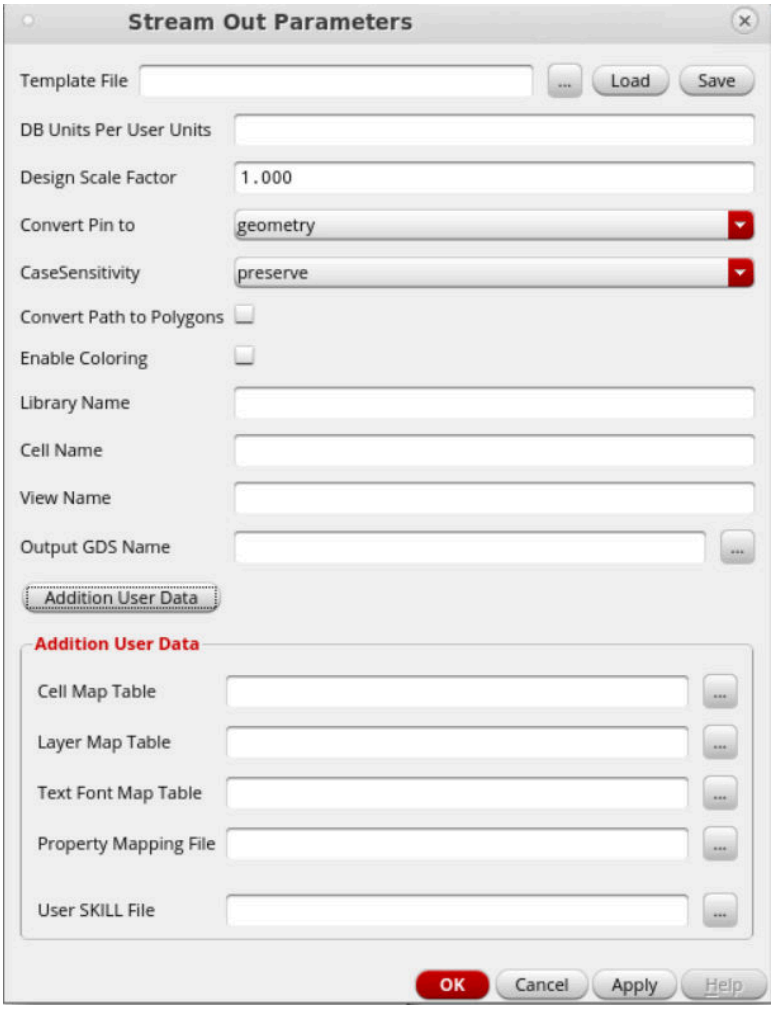
Figure 58 GDSII Stream Out Settings



Click **Addition User Data** in the Stream Out Parameters window to update mapping table (Figure 59).

Feedback

Figure 59 Updating Map Table in Addition User Data

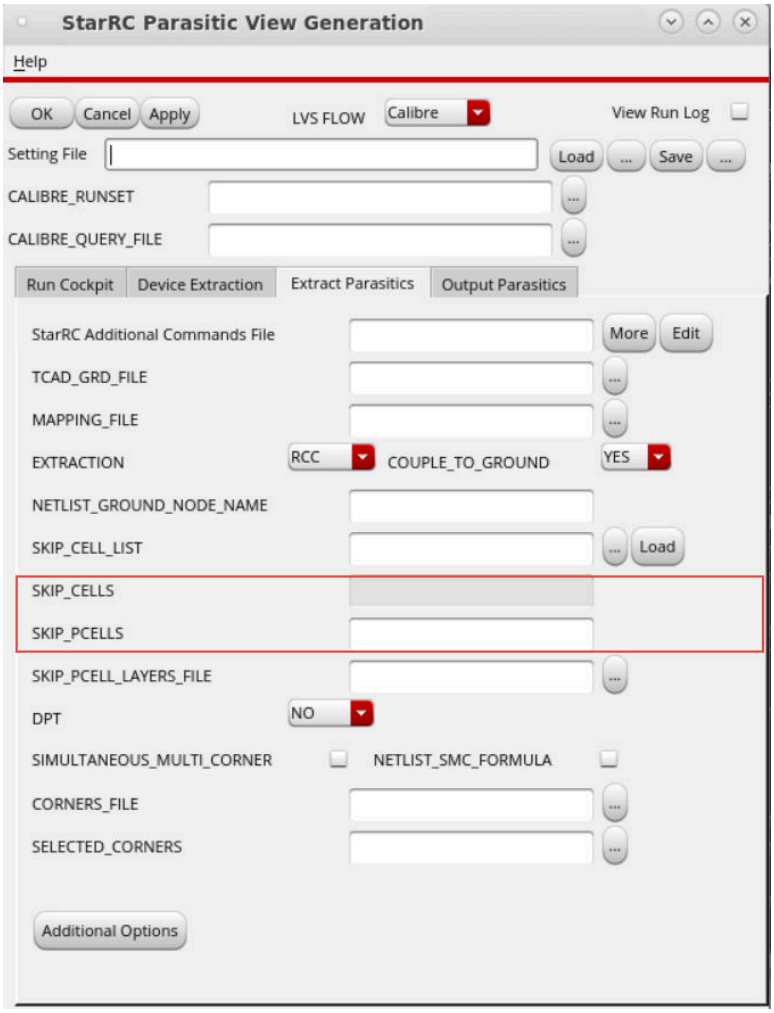


Feedback

Extract Parasitics Tab

Figure 60 shows the Extract Parasitics tab.

Figure 60 Extract Parasitics Tab



Extraction Options

The extraction options are shown in Table 39. If you select the RCC or RCG options, the `COUPLE_TO_GROUND` field shown in Figure 60 does not appear because the extraction option determines its setting.

Table 39 Extraction Options

RCC	RCG	NORC
EXTRACTION: RC COUPLE_TO_GROUND: NO	EXTRACTION: RC COUPLE_TO_GROUND: YES	NETLIST_SELECT_NETS: !* NETLIST_INSTANCE_SECTION: YES EXTRACTION: C

SKIP_CELL_LIST

The skip cell list is used for the Virtuoso Integration skip cell flow. The file format is the same as the device mapping file format, which is different from the standard StarRC skip cell file format. For more information, see The Device Mapping File. The following lines are examples of the correct file format:

```
skip_cell1 lib1 cell1 view1
 skip_cell2 lib2 cell2 view2
```

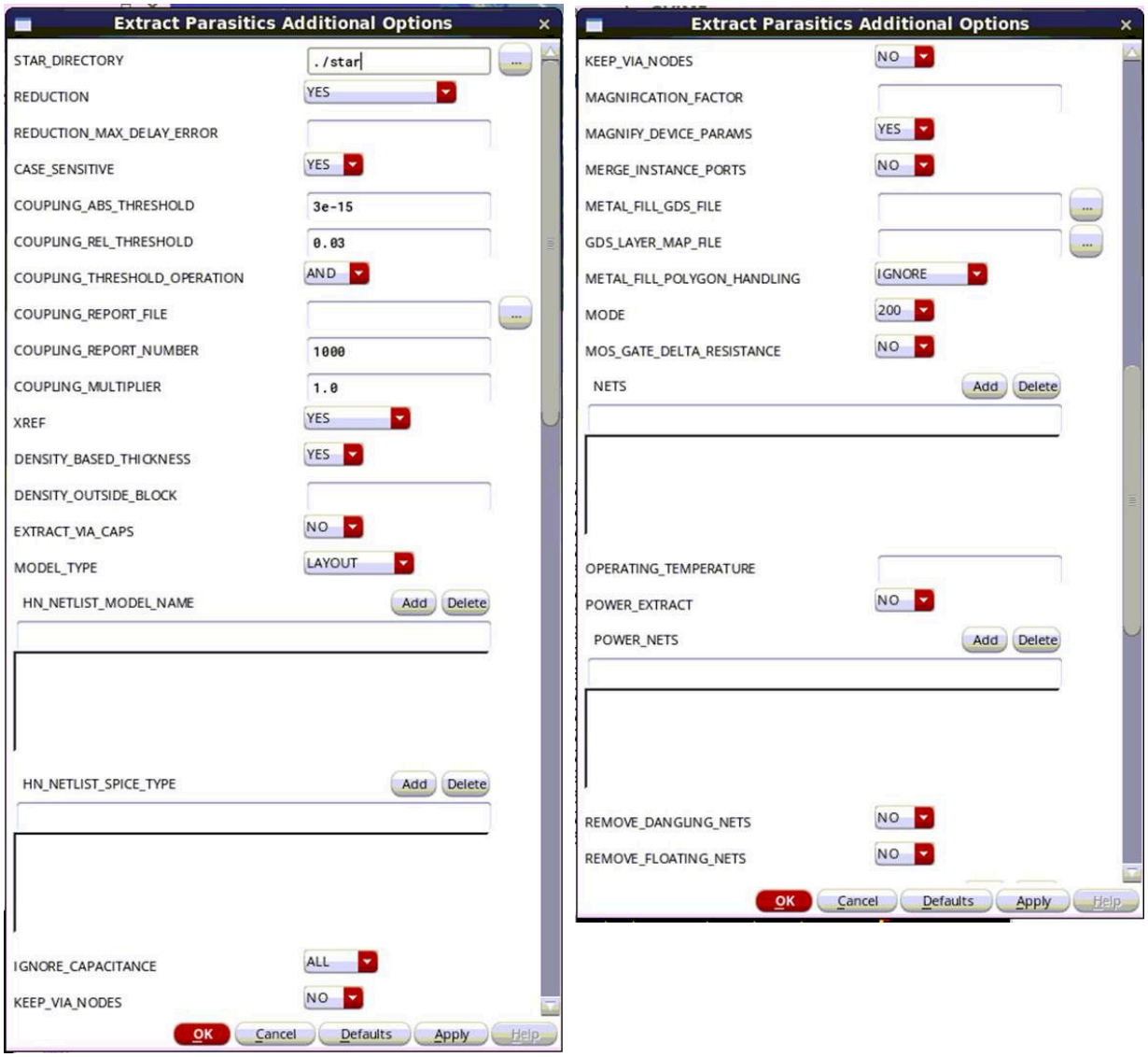
Additional Command File and Additional Options Dialog Box

For help in creating the many option combinations needed by Virtuoso Integration for view creation, you can add a command for the Cockpit and also adjust some options with the Additional Options dialog box.

Figure 61 and Figure 62 show the Additional Options dialog box, which is divided into three sections due to its length. For more information about the StarRC commands in this dialog box, see the command reference pages in Chapter 14, StarRC Commands.

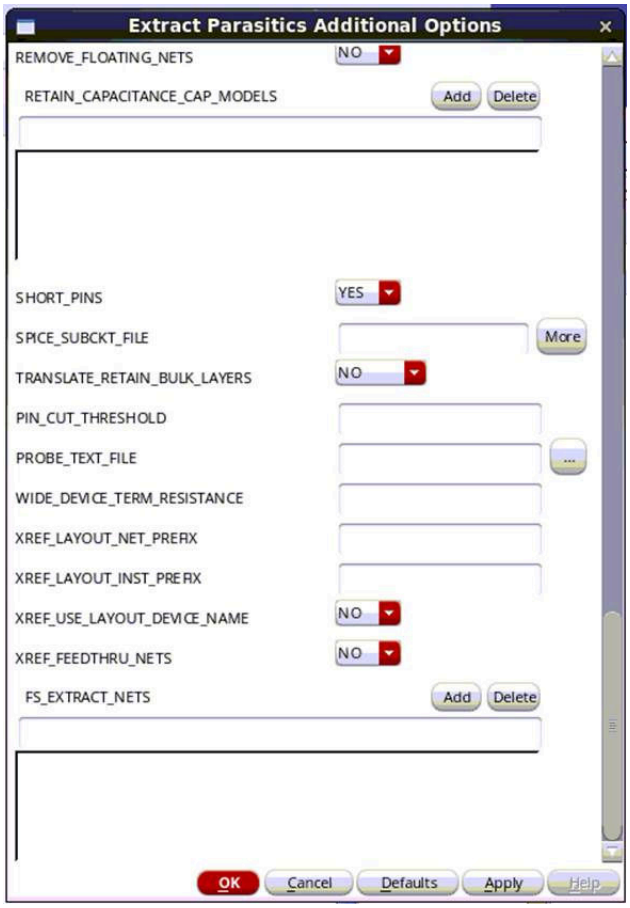
Feedback

Figure 61 Additional Options Dialog Box (Top and Middle Sections)



Feedback

Figure 62 Additional Options Dialog Box (Bottom Section)

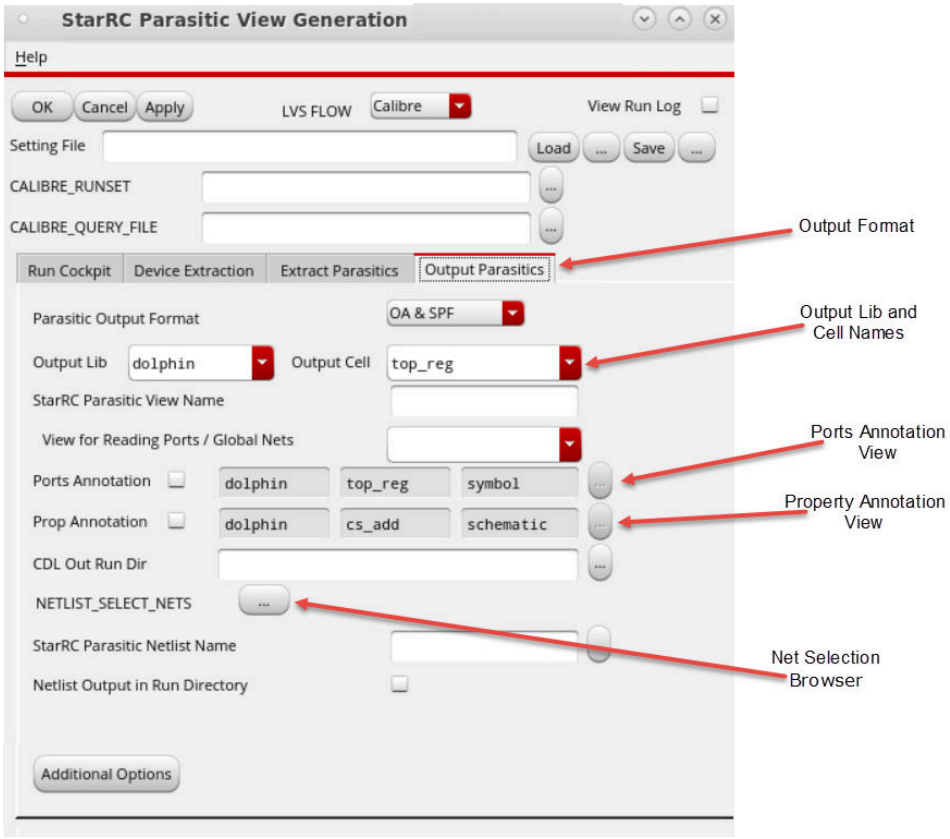


Feedback

Output Parasitics Tab

Figure 63 shows the Output Parasitics tab. You can use Virtuoso Integration as a GUI to configure the StarRC run.

Figure 63 Output Parasitics Tab

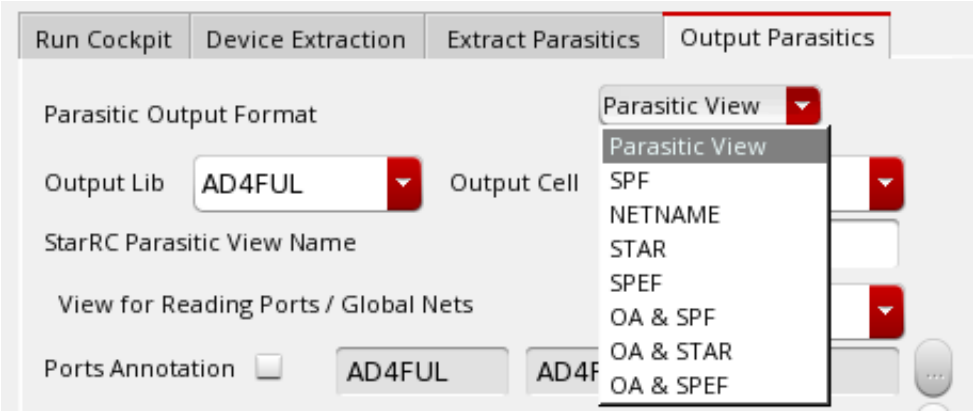


Parasitic Output Format

Use the pulldown list to select the output format, as shown in Figure 64. You can save a netlist in SPF, SPEF, or STAR format in addition to the OA parasitic view.

Feedback

Figure 64 Output Format Selection



Output Lib and Output Cell Names

By default, you cannot change the Output Lib and Output Cell names in the Cockpit. This default behavior prevents accidentally writing a newly-generated view to a previous cell from which the .snps_settings file was created.

To override this default behavior and save the Output Lib and Output Cell names to the .snps_settings file, set the `RC_SAVE_ALL` environment variable:

```
$ setenv RC_SAVE_ALL NO
```

Ports Annotation

Use this option when you are not generating a parasitic view as the top-block view, but want to integrate the parasitic view into other testbench circuits. Use the Ports Annotation View option to annotate the correct port and port direction list to the parasitic view. Then the parasitic view can be connected to the other view to form a complete circuit for simulation.

Property Annotation

Use this option to specify the view from which to obtain property information for schematic property annotation. This option defaults to the schematic view in the same library or cell window that starts the Virtuoso Integration Cockpit window.

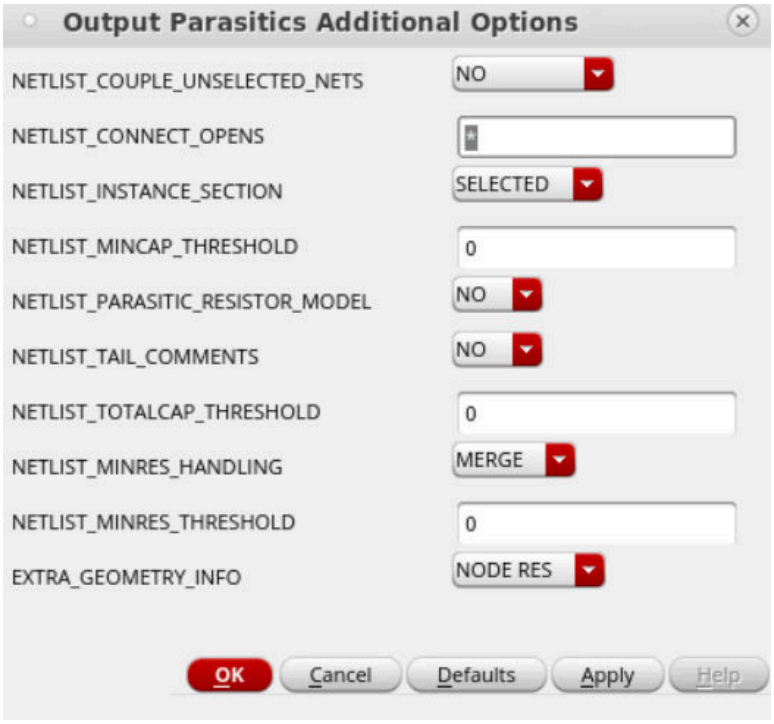
Netlist Selection Browser

Use this option to select nets to include in the netlist.

The Additional Options button brings up the dialog box shown in Figure 65. These options affect the output netlist.

Feedback

Figure 65 Output Parasitics Additional Options



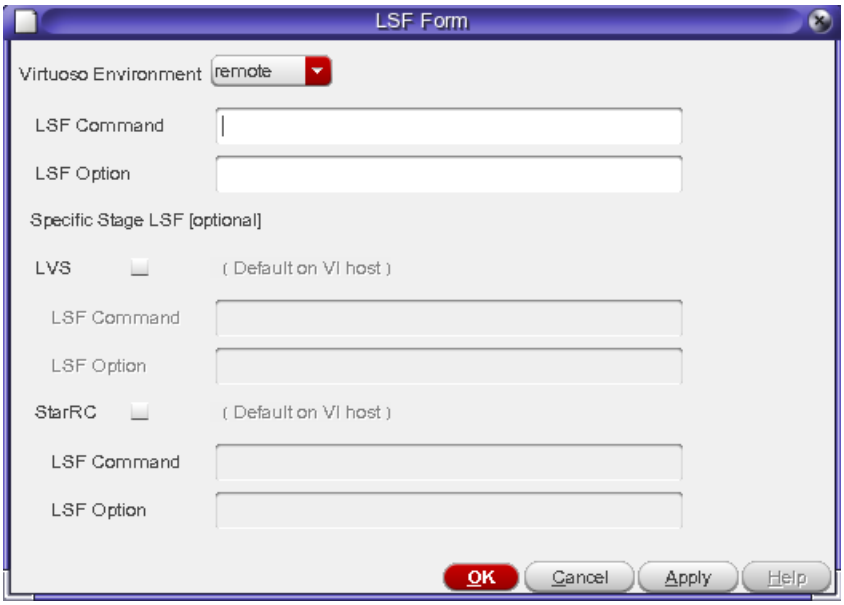
Load Sharing Facility Job Submission

Load Sharing Facility (LSF) support gives you the flexibility to control jobs that are submitted to specified farms. By default, all jobs, including LVS and StarRC extraction, are performed on the same remote server. However, if you want to run the LVS tool and the StarRC tool on different farms, use the LSF Form dialog box to specify LSF settings for each LVS and extraction task.

Specify whether to use LSF in the Run Cockpit tab with the Submit On LSF check box. If you check the check box, a button for the LSF settings form dialog box appears. An example of the form is shown in Figure 66. The LSF Form dialog box changes based on the flow selection.

Feedback

Figure 66 LSF Form Dialog Box



If you want to source an environment file before calling the StarRC and LVS jobs, such as setup license and path variables, you can create a wrapper to source these files first. The following is a simple example:

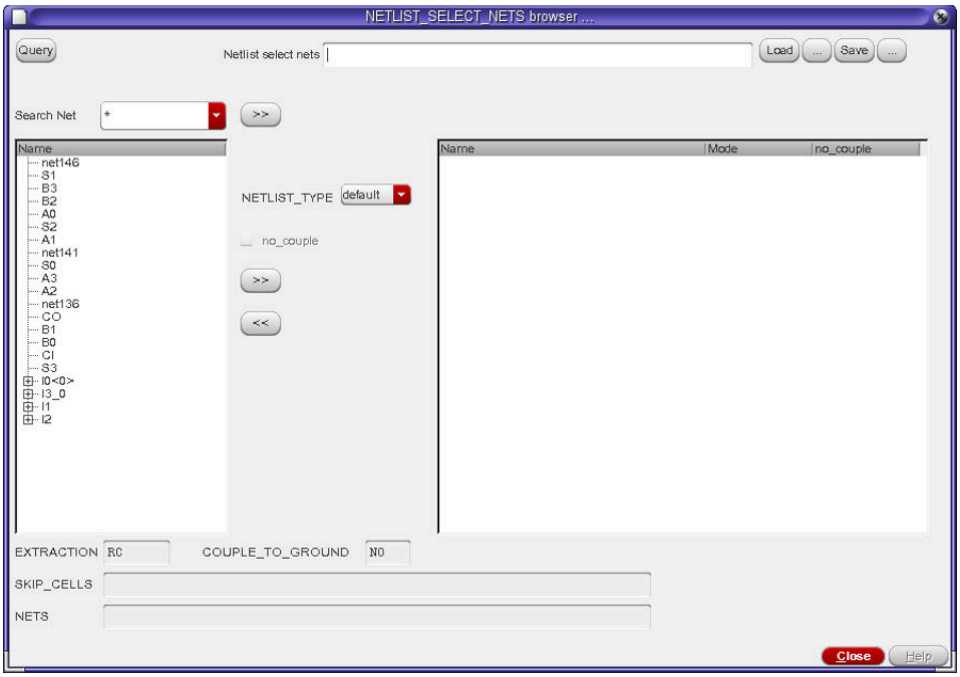
```
#!/bin/csh -fb
foreach arg ( $* )
  if( "$arg" == "-source" ) then
    set read_source = 1
    continue;
  endif

  if( $read_source == 1 ) then
     set source_file = $arg
     set read_source = 0
     continue;
  endif

  set args = "$args $last_arg"
  set last_arg = $arg
end

cat $source_file > tmp_file
echo $arg >> tmp_file

echo qsub $args tmp_file
$qsub $args $tmp_file
```

Feedback

Selecting Nets for Reporting

By default, the StarRC tool extracts and reports parasitics for all signal nets according to the settings of the EXTRACTION and COUPLE_TO_GROUND commands. However, you can limit the output to specific nets or specific types of parasitics by using the NETLIST_SELECT_NETS browser, shown in Figure 67.

Figure 67 NETLIST_SELECT_NETS Browser



The NETLIST_SELECT_NETS browser reads nets from the schematic. If the XREF command is set to NO during extraction, the NETLIST_SELECT_NETS browser is disabled. In this case, use the NETLIST_SELECT_NETS command instead by including it in an additional StarRC command file specified in the VI Cockpit.

Features of the NETLIST_SELECT_NETS browser are as follows:

- Extracted nets appear in the left window; instances appear as subtrees. The list of extracted nets is affected by the settings of the NETS and SKIP_CELLS commands that were used during extraction. The values are displayed at the bottom of the browser window for reference only.
- You can filter the list of nets by entering a string in the NETS field at the bottom of the browser window. Wildcards *, ?, and ! are acceptable. In Figure 68 part (a), the list of nets is restricted to nets beginning with the letter A.

- You can list all nets in the design by selecting “Show all nets.” In this case, the NETS field is ignored. Figure 68 part (b) illustrates the effect.

  Entering * in the NETS field or leaving it blank both mean to show all nets. In these cases, the “Show all nets” check box and label are hidden.

- Select specific nets by moving them from the left window to the right window using the button labeled “>>.”

  Before moving the selected nets, you can choose which parasitics to report by using the NETLIST_TYPE pulldown list; the choices are limited by the extraction that was performed. If the `COUPLE_TO_GROUND` command was set to `NO` during extraction and the selected netlist type is R, CG, or RCG, the check box labeled “no_couple” is available.

- You can also select nets by using the Search Net field; use spaces to separate multiple specifications. Wildcards *, ?, and ! are acceptable. Click the button labeled “>>” to select the nets displayed in the search results list. Figure 69 illustrates this feature.

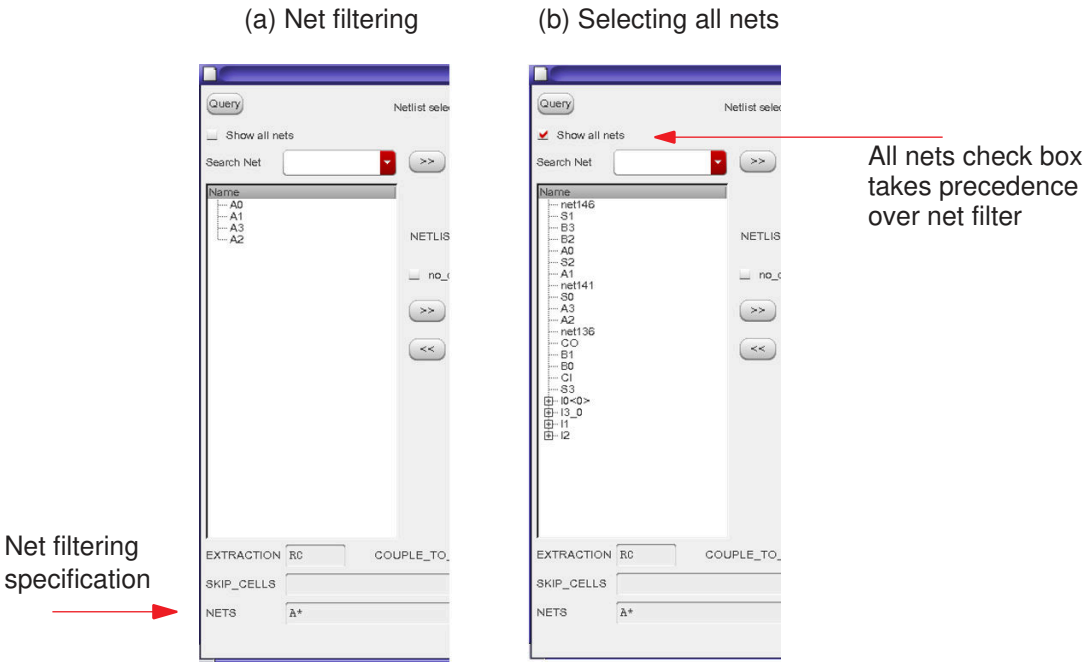
  Note:

  The drop-down list displays only schematic nets. However, the search function also returns layout-only nets. Using the “>>” button to select nets after a search might also select layout-only nets that meet the search specifications.

- If you try to select unextracted nets for output, an error message appears.
- If the same net is selected by multiple actions, the last action overrides all previous selection operations for that net.
- Use the shift+click key sequence to select more than one net, as illustrated in Figure 70.
- Query the schematic view by using the button in the top left corner. The queried net is highlighted in the left window.
- Load or save a file containing selected nets by using the buttons in the top right corner.

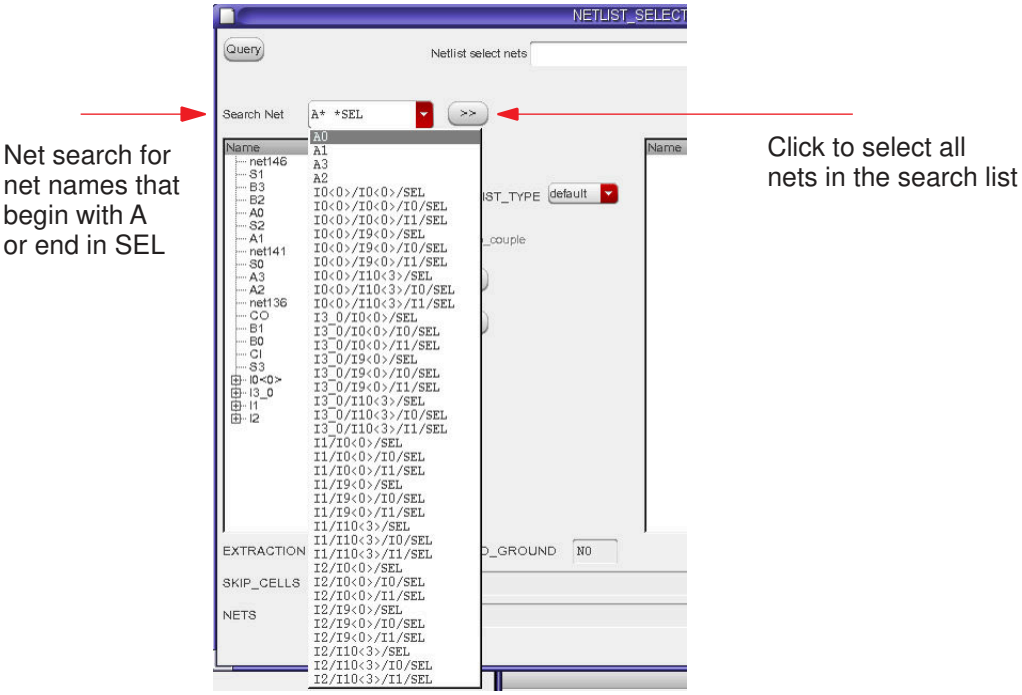
Feedback

Figure 68 Net Filtering



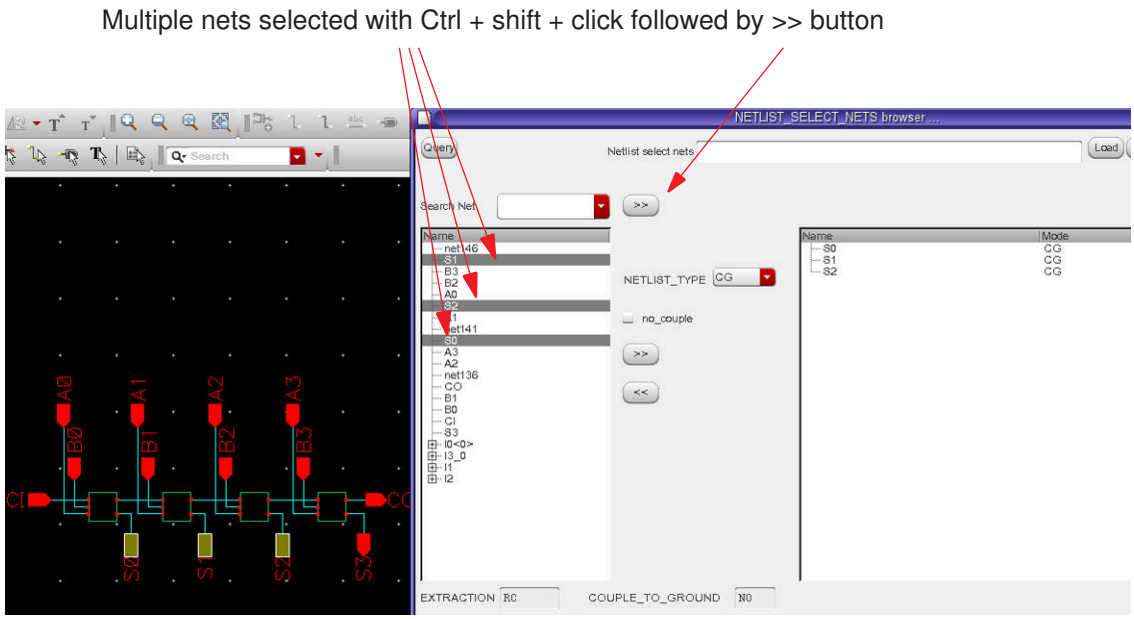
Feedback

Figure 69 Net Searching



Feedback

Figure 70 Multiple Net Selection



Feedback

Selecting and Customizing the Analysis Options

In the Run Cockpit dialog box, you can select Analysis options such as Temperature-VX, FieldSolver, electromigration analysis, or customized settings, as shown in Figure 71.

Figure 71 Analysis Options in Run Cockpit Dialog Box

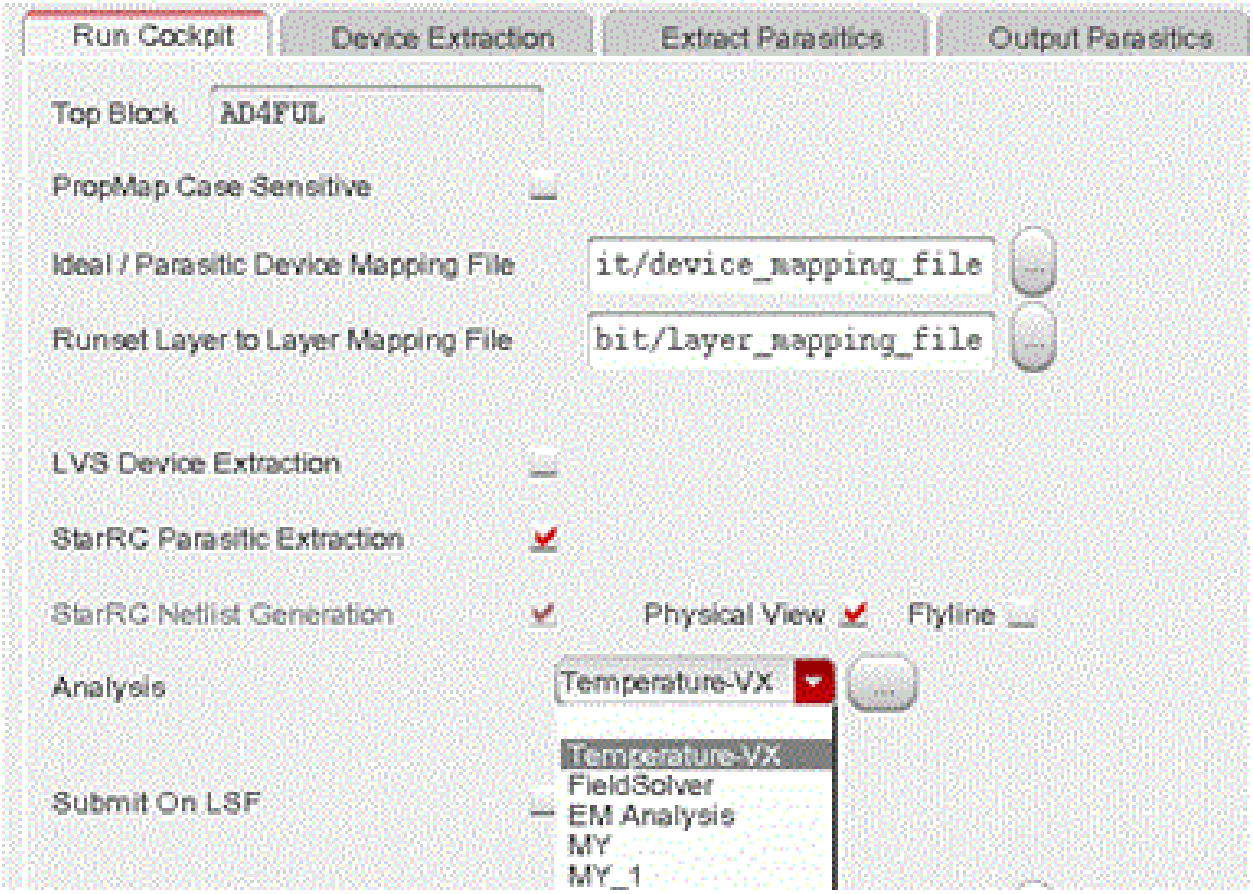


Table 40 describes the four predefined Analysis options.

Table 40 Predefined Analysis Options

Analysis option	Function
(blank)	Uses your current settings; this is the default setting
Temperature-VX	Allows you to output a single view with a selected corner
FieldSolver	Specifies the StarRC `FS_EXTRACT_NETS` command

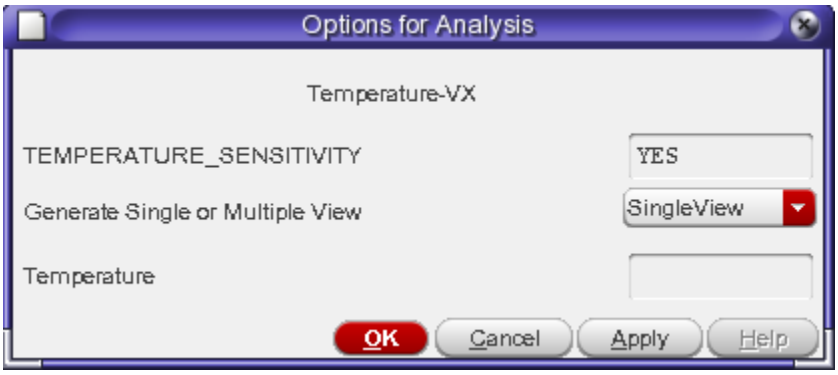
Feedback

Table 40 Predefined Analysis Options (Continued)

Analysis option	Function
EM Analysis	Specifies the StarRC `REDUCTION: NO`, `EXTRA_GEOMETRY_INFO: NODE RES`, and `POWER_REDUCTION: NO` commands

To store or edit the StarRC settings for the extraction run, select the analysis option in the Run Cockpit tab and click the "…" button. The Options for Analysis dialog box appears. An example is shown in Figure 72 for temperature sensitivity analysis.

Figure 72 Options for Analysis Dialog Box



To customize the StarRC settings for one or more analysis options,

1. Add the following statement to the .snps_settings file:

   ```
   ANALYSIS_SETTING: analysis_settings_file_name
   ```

2. In the analysis settings file, specify the `ANALYSIS` statement for each analysis setting followed by its corresponding StarRC commands. Use the following syntax:

   ```
   ANALYSIS: [" "] | Simulation | EM Analysis | FieldSolver |
             Temperature-VX | custom_setting_name
   StarRC_Command_1
   [StarRC_Command_2]
   …
   ```

To customize the default settings, specify

```
ANALYSIS: " "
```

The following example defines custom settings named ANALYSIS_CC and ANALYSIS_CG:

```
ANALYSIS: ANALYSIS_CC
EXTRACTION: RC
COUPLE_TO_GROUND: NO
```

```
ANALYSIS: ANALYSIS_CG
EXTRACTION: C
COUPLE_TO_GROUND: YES
```

StarRC OA View Creation

The OpenAccess view is a parasitic view that can be used directly in the Custom Compiler and Virtuoso design environments. This view includes all parasitic components and network connections along with physical polygons.

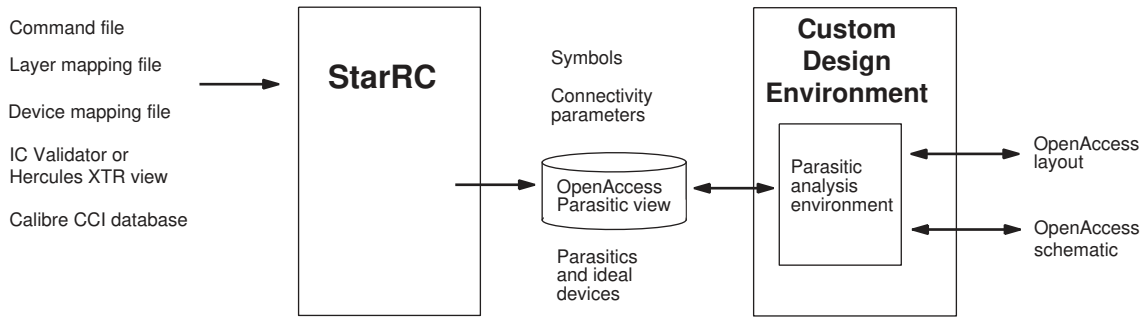
The StarRC tool can generate OpenAccess format parasitic views outside the custom design environment. However, the Custom Compiler GUI or Virtuoso Integration Cockpit GUI are helpful for setting up many options.

The OpenAccess Flow

Figure 73 illustrates the OpenAccess flow in the custom design environment. You can run extraction and generate a parasitic view within the design environment for efficient post-layout simulation. The parasitic view provides you with an accurate representation of the parasitics and is generated using either SKILL-based functions or native functionality.

Accurate layout representation requires netlist connectivity information to instantiate each extracted parasitic element and design device. The parasitic view provides you with an efficient and detailed analysis tool in the physical domain. The parasitic view must contain schematic symbols that can be written in the netlist for simulation as well as used as a graphical tool for identifying parasitic devices.

Figure 73 OpenAccess Flow



Support for Special StarRC Flows

The StarRC simultaneous multicorner flow is supported for OpenAccess views.

The temperature sensitivity flow is also supported for the creation of multiple OA views.

Feedback

Skip Cell Mapping

If you use skip cells in the Virtuoso Integration flow, the number of ports and the port names in the OpenAccess symbol view of the skip cell must match the number of ports and the port names created from the layout during layout versus schematic (LVS) checking done before parasitic extraction.

If the number of ports and the port names match, use the `OA_SKIPCELL_MAPPING_FILE` command to specify a file that defines which cell master to use for the skip cells defined in the `SKIP_CELLS` command. The syntax is as follows:

```
INV1 myLib INV symbol
```

If there is a mismatch in either the number of ports or the port names, use the following guidelines to ensure that skip cells are properly connected in the OpenAccess view created by the StarRC tool:

- The number of ports is different, but port names match

  Use the `SPICE_SUBCKT_FILE` command to specify SPICE files that contain .subckt definitions for the skip cells. The StarRC tool uses the schematic ports found in the SPICE files instead of the LVS port definitions.

  In addition, use the `OA_SKIPCELL_MAPPING_FILE` command to specify a file that defines which cell master to use for the skip cells defined in the `SKIP_CELLS` command.

- The number of ports matches, but some or all port names are different

  Use the `OA_DEVICE_MAPPING_FILE` command to specify a file that contains skip cell mapping definitions. An example of the syntax is as follows:

  ```
  mimcap myLib mimcap symbol PLUS MINUS SHIELD1 SHIELD2
  ```

- The number of ports and the port names are both different

  You can define both the number of ports and the port names in the device mapping file (specified with the `OA_DEVICE_MAPPING_FILE` command).

  Alternatively, you can define the number of ports with SPICE .subckt files (specified with the `SPICE_SUBCKT_FILE` command) and the port names with a device mapping file (specified with the `OA_DEVICE_MAPPING_FILE` command).

Feedback

OpenAccess File Examples

The following sections provide examples for the OpenAccess library definition and the layer and device mapping files.

OpenAccess Library Definition

The following is an example of the OpenAccess library specified by the `OA_LIB_DEF` command in the StarRC command file:

```
DEFINE w_xxb_fifo1 /remote/cae933/VI/OA/w_xxb_fifo1
DEFINE N90lo /testcases/misc/star_virtuoso/N90lo
DEFINE cdsDefTechLib /global/apps/ic_61/linux/tools/dfII/etc/
cdsDefTechLib
DEFINE analogLib /global/apps/ic_61/linux/tools/dfII/etc/cdslib/artist/
analogLib
DEFINE US_8ths /global/apps/ic_61/linux/tools/dfII/etc/cdslib/sheets/
US_8ths
DEFINE basic /global/apps/ic_61/linux/tools/dfII/etc/cdslib/basic
```

OpenAccess Mapping Files

The following is an example of the OpenAccess layer mapping file specified by the StarRC `OA_LAYER_MAPPING_FILE` command:

```
poly     PO drawing PO pin PO dummy
metal1   M1 drawing M1 pin M1 dummy
metal2   M2 drawing M2 pin M2 dummy
metal3   M3 drawing M3 pin M3 dummy
metal4   M4 drawing M4 pin M4 dummy
metal5   M5 drawing M5 pin M5 dummy
metal6   M6 drawing M6 pin M6 dummy
metal7   M7 drawing M7 pin M7 dummy
Cont  CO drawing nil nil nil nil
pl3co   CO drawing nil nil nil nil
VIA1    VIA1 drawing nil nil nil nil
VIA2    VIA2 drawing nil nil nil nil
VIA3    VIA3 drawing nil nil nil nil
VIA4    VIA4 drawing nil nil nil nil
VIA5    VIA5 drawing nil nil nil nil
VIA6    VIA6 drawing nil nil nil nil
```

The following is an example of the OpenAccess device mapping file specified by the StarRC `OA_DEVICE_MAPPING_FILE` command:

```
pres analogLib presistor auLvs PLUS MINUS
pcap analogLib pcapacitor auLvs PLUS MINUS
nch N90lo nch auLvs G S D B
pch N90lo pch auLvs G S D B
nch_18 N90lo nch_18 auLvs G S D B
```

Feedback

StarRC Commands for OpenAccess Parasitic Views

The StarRC commands that affect the OpenAccess flow are listed in Table 41.

Table 41 Commands for OpenAccess Flow

Command	Type	Default	Description
NETLIST_FORMAT	string	NETNAME	Set to OA; required
OA_BUS_BIT	string	Same as BUS_BIT	Specifies the bus bit delimiter
OA_CDLOUT_RUNDIR	string	none	Directory that contains the ihnl subdirectory and mapping files
OA_CELL_NAME	string	none	OpenAccess cell name
OA_DEVICE_MAPPING_FILE	string	none	File that describes device mapping
OA_INSTANCE_PIN_NAME	string	SUBCKT	Specifies where to get pin names
OA_LAYER_MAPPING_FILE	string	none	File that describes layer mapping
OA_LIB_DEF	string	lib.defs	OpenAccess library definition file; optional
OA_LIB_NAME	string		OpenAccess library name
OA_MARKER_SIZE	float	0.1	Port or subnode marker size (microns); optional
OA_MULTI_OUTPUT	string	SPF	Format of parasitic netlist to generate in addition to OA parasitic view
OA_OVERWRITE_LOCKED_VIEW	Boolean	NO	Allows the StarRC tool to overwrite a locked parasitic view
OA_PORT_ANNOTATION_VIEW	string	null string	Enables the simulation of a parasitic view; generated by the OpenAccess writer
OA_PROPERTY_ANNOTATION_VIEW	string	none	Specifies which schematic library, cell, or view is used to check against ideal devices for schematic-only properties and to attach them into the OpenAccess parasitic view
OA_PROPMAP_CASE_SENSITIVE	Boolean	NO	Case sensitivity of property mapping
OA_REMOVE_DUPLICATE_PORTS	Boolean	NO	Prevents duplication of port names
OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX	Boolean	YES	Specifies whether to remove SPICE card prefix characters before primitives in ideal instance names

Feedback

Table 41 Commands for OpenAccess Flow (Continued)

Command	Type	Default	Description
`OA_REMOVE_SPICECARD_PREFIX`	Boolean	`YES`	Specifies whether to remove SPICE card prefix characters from instance name paths
`OA_SKIPCELL_MAPPING_FILE`	string	none	Specifies cell master to use for skip cells in the parasitic view
`OA_VIEW_NAME`	string	`starrc`	Parasitic view name

Parasitic Probing

After the parasitic view is generated, you can probe the parasitic view or the schematic view to understand the StarRC extraction results.

This section has the following topics:

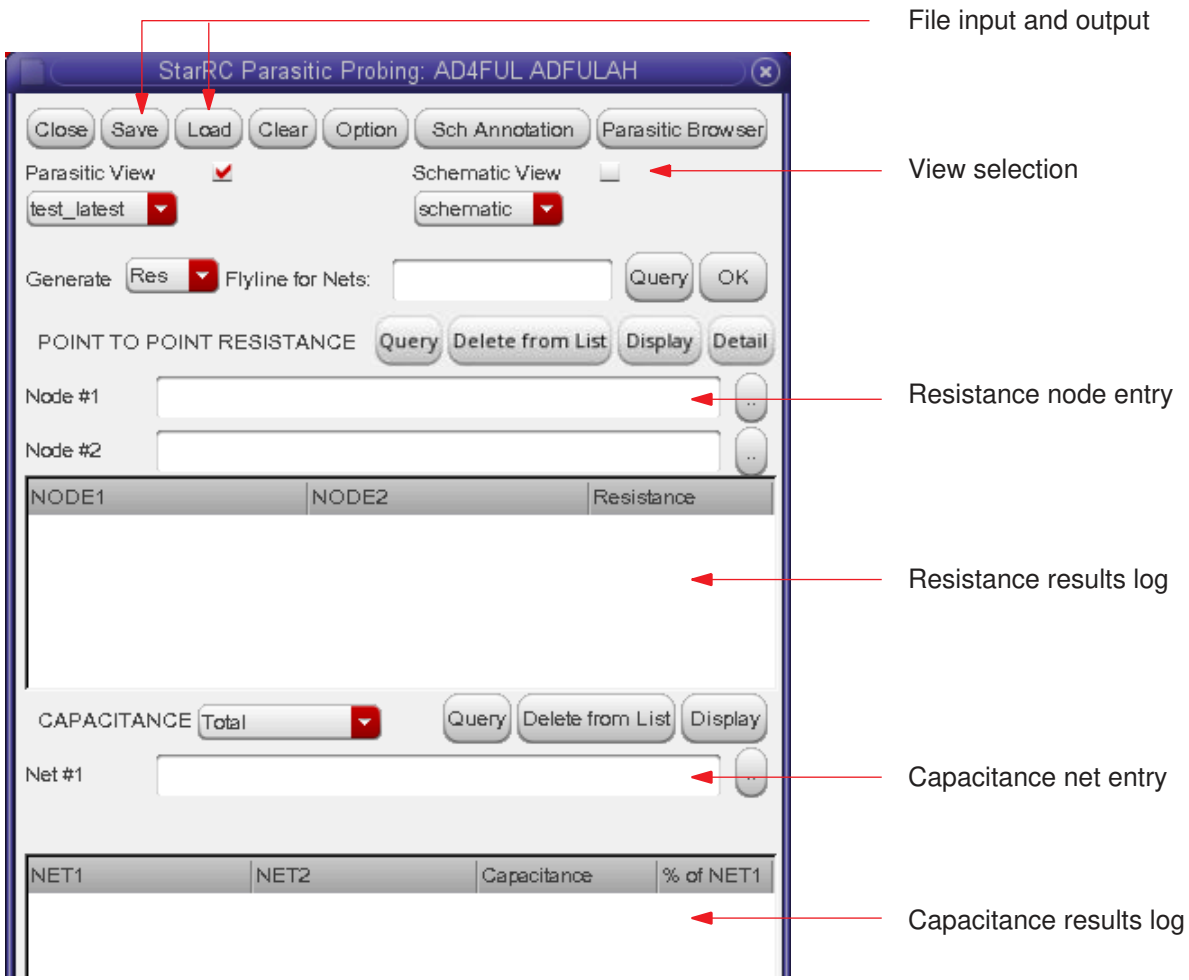
- StarRC Parasitic Prober
- StarRC Parasitic Browser
- StarRC Parasitic Netlist Browser
- StarRC Parasitic Explorer in the Virtuoso Tool
- View Selection
- Dynamic Flylines for Probing
- Point-to-Point Resistance Probing
- Probing a Single Bus Bit or All Bus Bits
- Displaying Multiple Nets
- Parasitic View Probing
- Schematic View Probing
- Probed Results Log and Cross-Probing
- Prober File Input and Output
- Schematic Annotation

Feedback

StarRC Parasitic Prober

The StarRC parasitic prober is available through the Parasitic Prober entry of the StarRC pulldown menu within any parasitic or schematic view window. Use the StarRC Parasitic Probing dialog box, shown in Figure 74, to probe parasitics within the parasitic view or the corresponding schematic view.

Figure 74 StarRC Parasitic Probing Dialog Box



Choose one of the following probing modes:

- Parasitic View
  - Probes port and subnode markers for point-to- point resistance between any two same-net points.
  - Probes interconnect polygons for total net capacitance, with or without couplings to constituent nets.
  - Probes interconnect polygons for total coupling capacitance between two nets.
- Schematic View
  - Probes schematic instance terminals for point- to-point resistance between any two same-net terminals, at any level of hierarchy contained within the schematic cell matching the extracted cell.
  - Probes schematic nets for total net capacitance, with or without couplings to constituent nets, at any level of hierarchy contained within the schematic cell matching the extracted cell.
  - Probes schematic nets for total coupling capacitance between two nets, at any level of hierarchy contained within the schematic cell matching the extracted cell.

The prober also provides the following additional features:

- Highlighting and zooming of parasitic view interconnect polygons corresponding to a previously probed total capacitance or point-to-point resistance result.
- Annotation of total capacitance results to a corresponding schematic view window
- Sorting of logged resistance and capacitance results based on net name or parasitic value
- Output of probed parasitic results to an ASCII report file, as well as input of parasitic results from a previously output ASCII report file
- Activation or deactivation of flylines representing individual parasitic resistors and capacitors

Feedback

StarRC Parasitic Browser

From the Parasitic Browser, you can query a net name. All the node connections and parasitic devices on this net are listed in the dialog box. You can review this information, select the node, resistor, or capacitor to be deleted or display information about the StarRC view.

Type a name, then click Search, and the Parasitic Browser displays all the net names that contain the input pattern. When the required name is shown, click Done. The Parasitic Browser parses the net to show all the physical nodes in the Connection field.

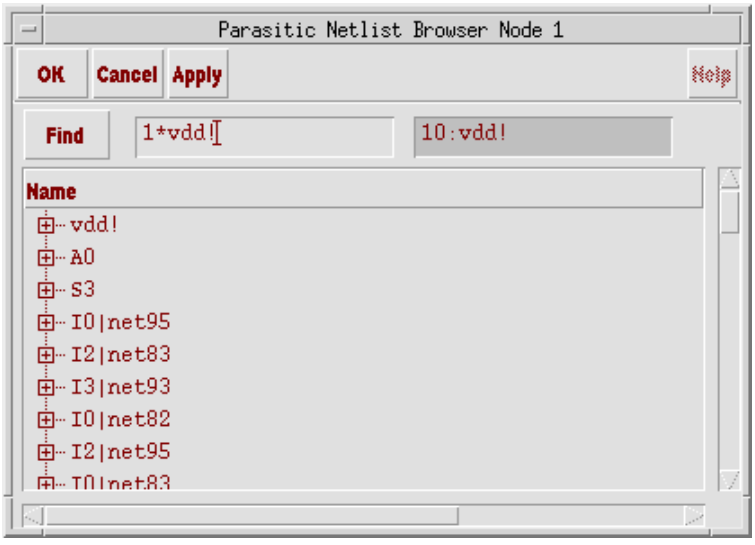
Table 42 Parasitic Browser Button and Field Functions

Function	Description
Query	Chooses a net and sends it to the Connection field by querying a name from schematic view or parasitic view
Display	Displays the selected node to be highlighted in the parasitic view
Delete	Deletes the selected node

StarRC Parasitic Netlist Browser

The Parasitic Netlist Browser, as shown in Figure 75, helps you find a specific node. Click the plus sign to the left of a net name to expand the net and show the signal group. Click the minus sign to collapse the signal group into a net.

Figure 75 Parasitic Netlist Browser



Feedback

Use the Parasitic Browser to browse, search, and select a net name to send to the prober. You can enter a net name with wildcard in the Find box. When you click Find, the matching net name appears. When you click Apply, the pattern is sent to the node or net field.

Note:

Only one wildcard is allowed in the search string. A string that contains more than one wildcard or the question mark (?) character might not return the expected results.

StarRC Parasitic Explorer in the Virtuoso Tool

The Virtuoso Integration (VI) interface allows you to perform the following tasks using the StarRC Parasitic Explorer tool:

- Highlight resistors and capacitors in an OA view
- GUI based analysis of parasitics and errors

For information to launch the Parasitic Explorer GUI from the Virtuoso StarRC Parasitic Prober and to use the Parasitic Explorer commands, see the *StarRC Parasitic Explorer User Guide* on SolvNetPlus.

See Also

- StarRC Parasitic Prober

View Selection

Parasitic view probing is done either within the parasitic view or the schematic view. The Parasitic View and Schematic View radio buttons at the top of the prober dialog box enable probing for either the selected parasitic view or the selected schematic view. Only one probing mode is selectable at a time, but the mode can be changed at any time using these radio buttons.

The menus beneath the Parasitic View and Schematic View radio buttons enable you to select the specific view names to be used for each mode. The parasitic view name or schematic view name can be changed at any time. Note that when parasitic view probing is in effect, the selected schematic view is not relevant and is ignored. However, when schematic view probing is in effect, the selected parasitic view specifies the view from which resistance and capacitance parasitics is read.

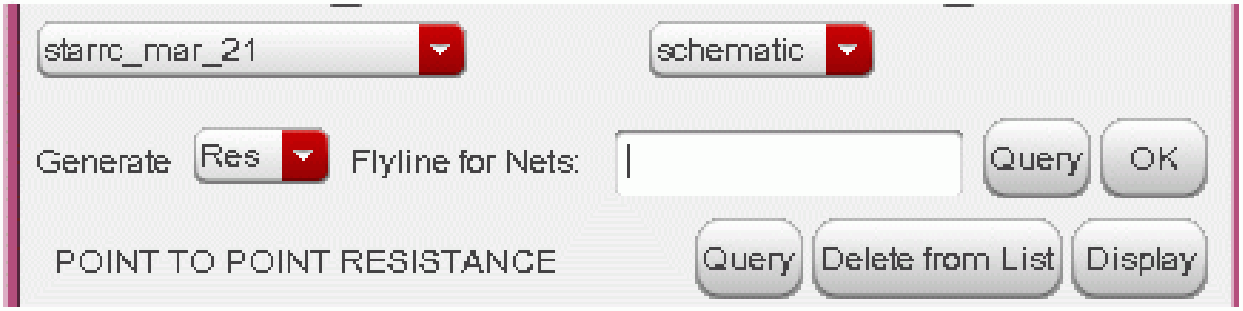
Feedback

Dynamic Flylines for Probing

When you want to probe point-to-point resistance or capacitance, flylines can help you select the right node pair. Virtuoso Integration has the capability to generate flylines dynamically only for certain nets.

In the Flyline for Nets field in the Prober GUI, shown in Figure 76, enter the net names or click Query to start a search. When you click OK, Virtuoso Integration generates the flylines for the specified nets.

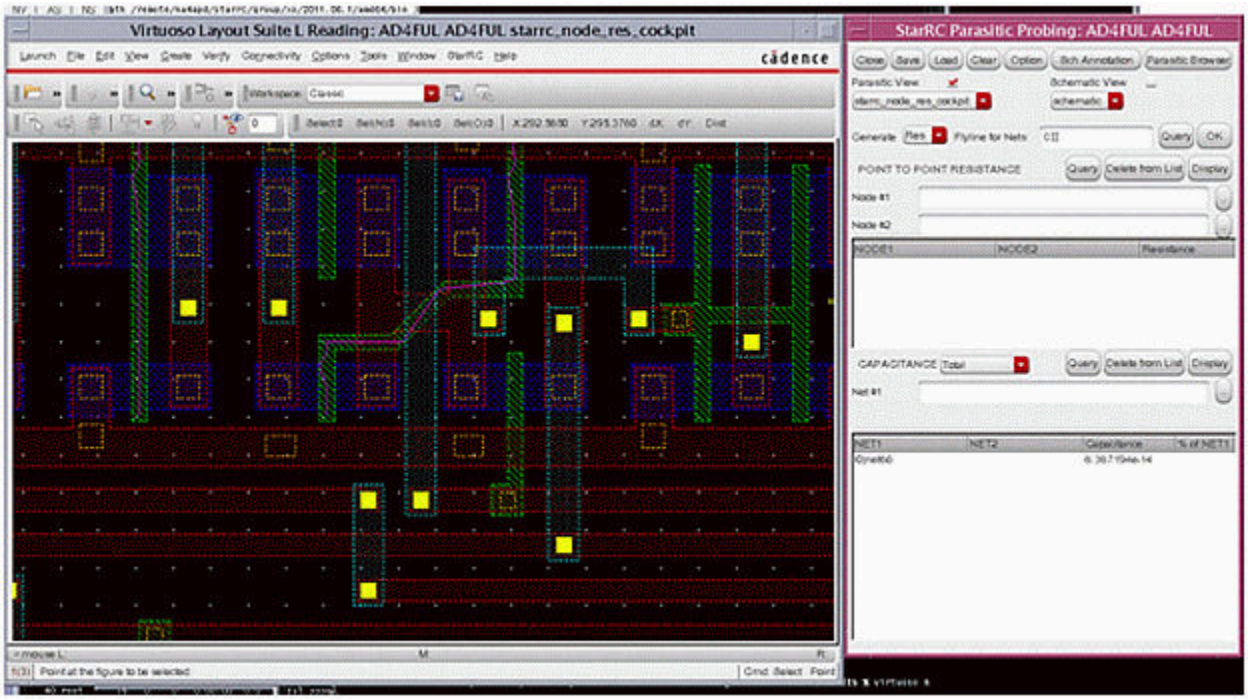
Figure 76 Specify Nets to Generate Dynamic Flyline



Feedback

The flylines can assist you in point-to-point probing. Figure 77 shows the flyline generated for net CI.

Figure 77 Dynamic Flyline Probing



Point-to-Point Resistance Probing

Point-to-point (P2P) resistance probing allows you to query two same-net nodes from the selected view and derive the corresponding parasitic path resistance between these two nodes. This calculation uses resistance network reduction techniques to reduce the network down to the two selected nodes and report the equivalent resistance between the two nodes.

In addition to probing nodes, you can also manually enter the node names into the corresponding text boxes in the prober dialog box to compute equivalent point-to-point resistance.

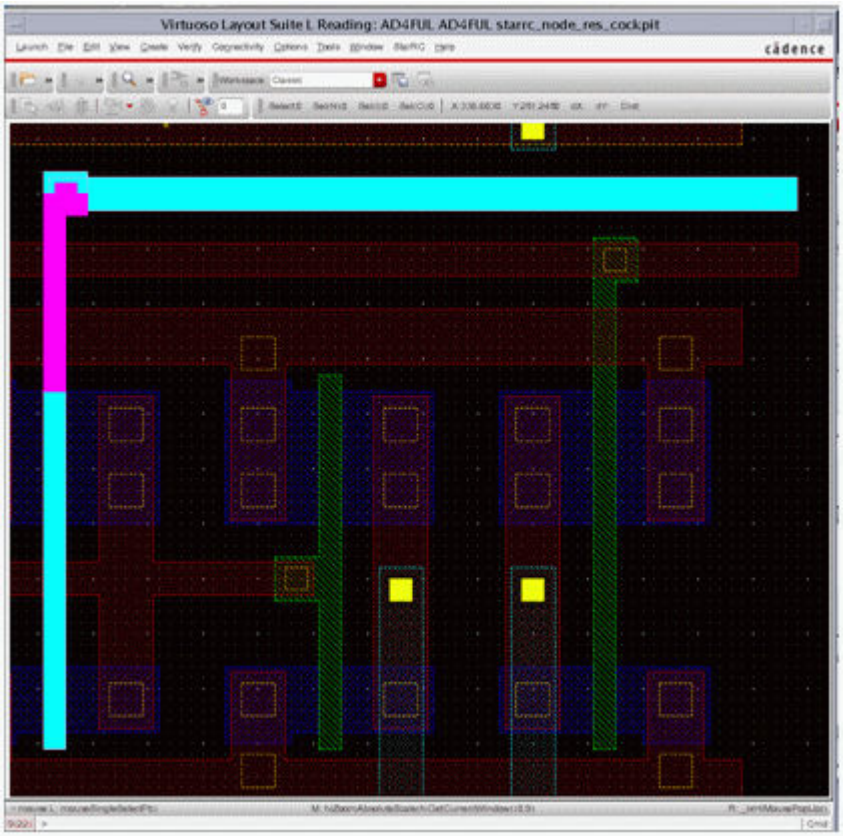
Double Highlighting of Point-to-Point Resistance Probe Results

The Virtuoso Integration interface can display double highlighting for the probe results of a point-to-point resistance network. This feature helps to visualize the parasitic extractions results more easily.

Feedback

Figure 78 shows an example of double highlighting. The entire net is highlighted in aqua. The point-to-point resistance probe results are highlighted in magenta.

Figure 78 Double Highlighting of a Point-to-Point Resistance Network



To enable double highlighting, add the following commands to the StarRC command file:

```
REDUCTION : NO
EXTRA_GEOMETRY_INFO: NODE RES
```

Alternatively, you can select EM_Analysis for the analysis type on the Run Cockpit tab. EM Analysis automatically sets the following commands:

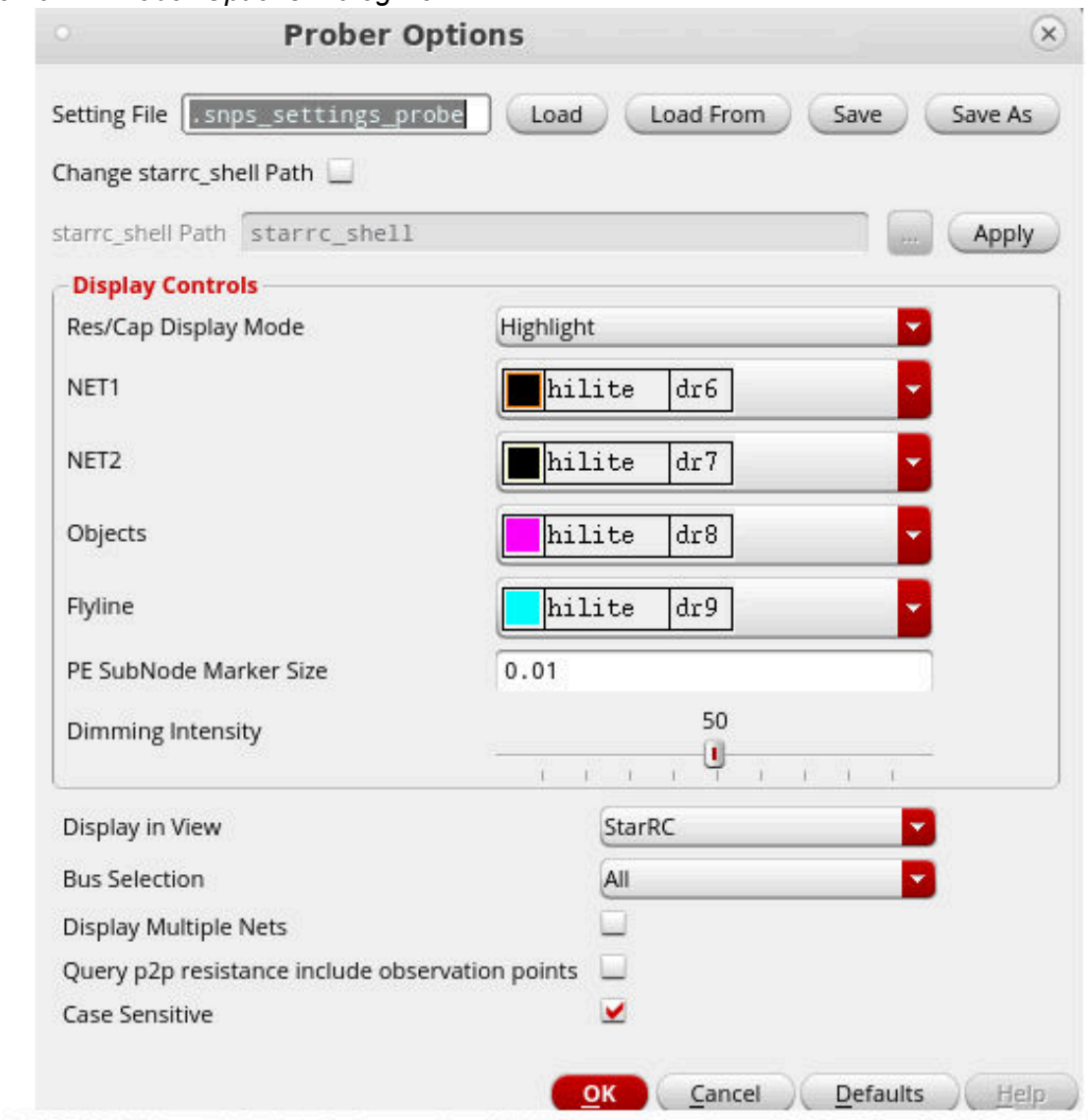
```
REDUCTION : NO
EXTRA_GEOMETRY_INFO: NODE RES
POWER_REDUCTION: NO
NETLIST_TAIL_COMMENTS: YES
```

Feedback

Specifying and Saving the Probe Options

To adjust the highlighting properties, click the Option button. The dialog box shown in Figure 79 appears.

Figure 79 Prober Options Dialog Box



To save the configuration, specify the following settings in the .snps_settings_probe file:

- `DISPLAY_IN_VIEW: STARRC | SCHEMATIC | STARRC_AND_SCHEMATIC`

  The default is `STARRC`. Specify `SCHEMATIC` or `STARRC_AND_SCHEMATIC` to send highlights to the schematic view.

- `BUS_SELECTION: ALL | SINGLE_BIT`

  The default is `ALL`, which selects the entire bus in a node or wire. If you specify the `SINGLE_BIT` option, the Select Bus Bit dialog box opens.

- `DISPLAY_MULTI_NETS: NO | YES`

  The default is NO; which specifies that a new highlight clears a previous highlight.

Highlighting or Blinking Probe Results

The Res/Cap Display Mode gives you two choices: Highlight and Blinking.

You can specify the color and fill of the highlights in the LPP settings boxes. The color and fill is picked up from your Virtuoso display resource.

If you set the Res/Cap Display Mode to Blinking, the Prober displays the probe results as blinking highlights. Note the following limitations to the blinking highlights:

- Only the highlight for the whole net blinks. The highlight of the P2P partial net does not blink.
- In blinking mode, the Prober changes the display resource to have blinking LPP. A new packet—with “B” appended to the original name—is created and used for this LPP.
- The prober tries to add temporary shapes to the view for blinking effects.

If you save the view, Virtuoso Integration asks you to confirm saving the changes made by the blinking mode. If you do not want to save the changes made by Virtuoso Integration, choose Cancel.

Viewing Details of Resistance and Capacitance

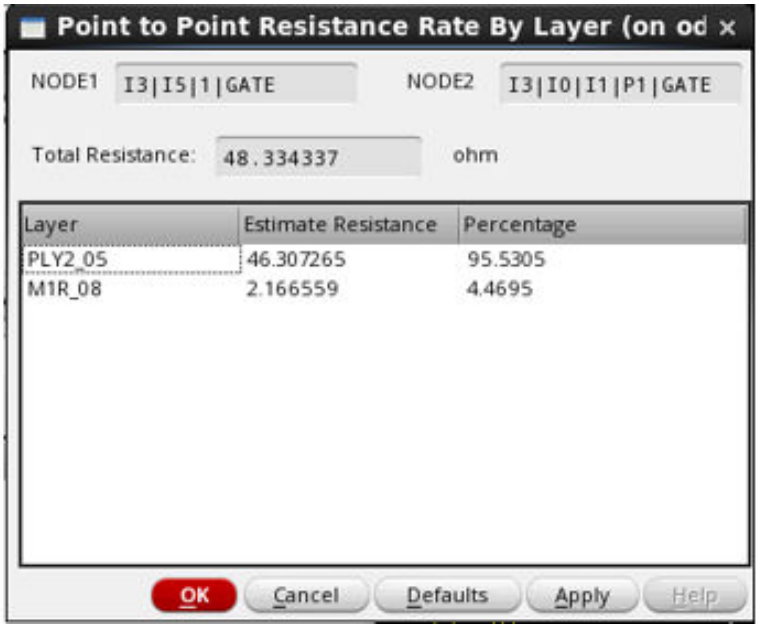
You can enable the StarRC parasitic prober to view detailed contributions of point-to-point resistance and net-to-net capacitance on each layer if a netlist contains the geometric information of pins, ports, and nodes.

To view details of point-to-point resistance and net-to-net capacitance for a layer, you must save the geometric information by using the following commands in the StarRC command file:

```
EXTRA_GEOMETRY_INFO: NODE RES
NETLIST_TAIL_COMMENT: YES
REDUCTION: NO
```

To view the contribution of point-to-point resistance and net-to-net capacitance on each layer, in the StarRC Parasitic Probing dialog box (Figure 74), click the Detail button. The window displays the following details:

- Computes point-to-point resistance contribution from each layer and shows point-to-point resistance ratio from node-to-node, total resistance, layer name, and estimate resistance and percentage (estimated), as shown in the following figure:



- Computes net-to-net capacitance or net total capacitance contribution from each layer and shows coupling capacitance of both nets, NET1 and NET2, total capacitance of

one net, layer name, and estimate capacitance and percentage (estimated), as shown in Figure 80 and Figure 81:

Figure 80 Net total capacitance

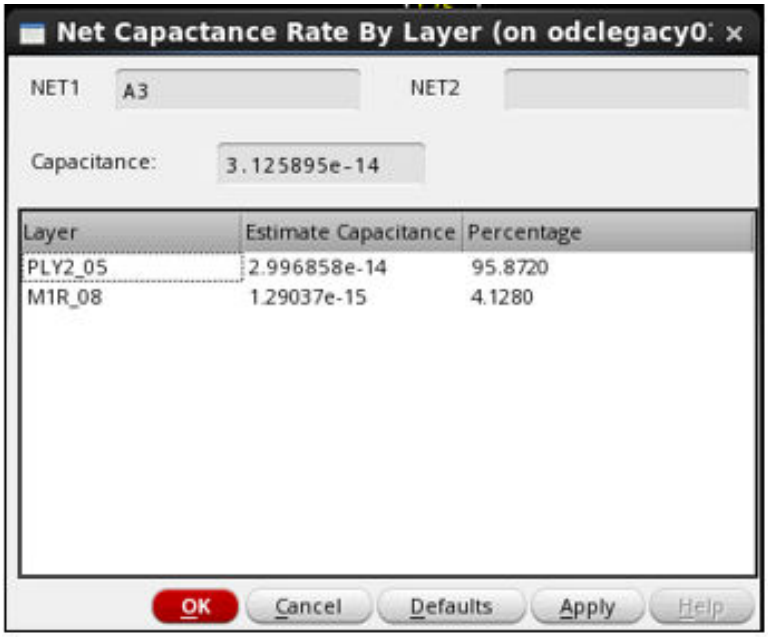
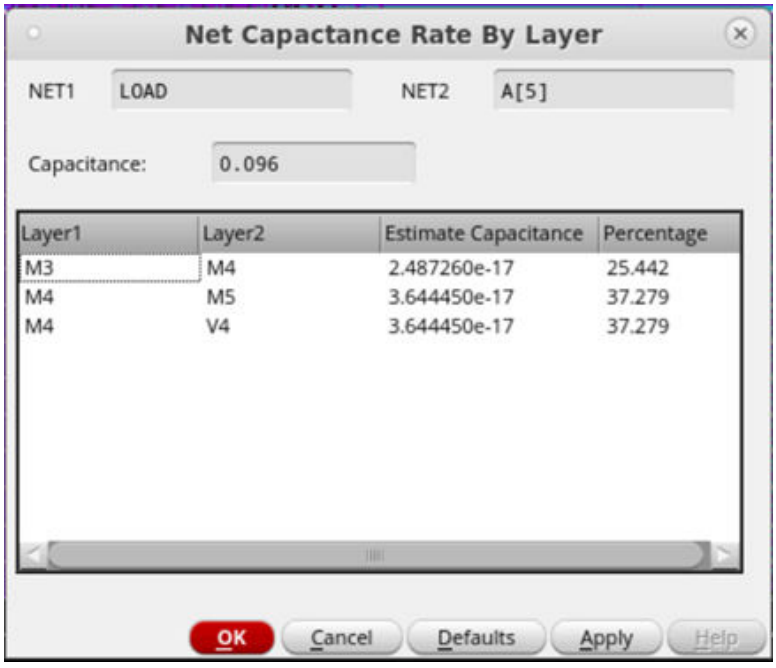


Figure 81 Net-to-net capacitance, that is coupling capacitance



Feedback

Probing a Single Bus Bit or All Bus Bits

You can choose to probe a single bus bit or all bits in a bus, as shown in Figure 82.

Figure 82 Bus Bit Selection in the Probe Options Dialog Box

Figure 83 shows an example of probing results when you choose to probe all bus bits.

Figure 83 Probing All Bus Bits

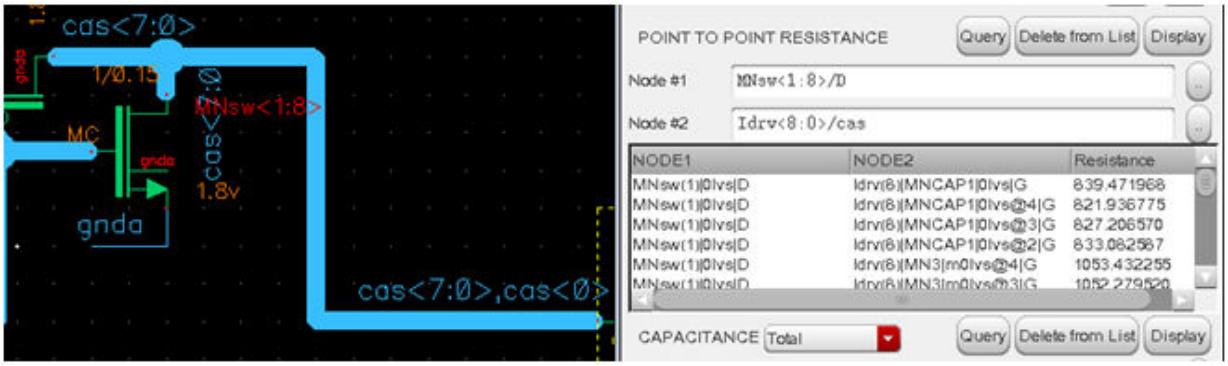
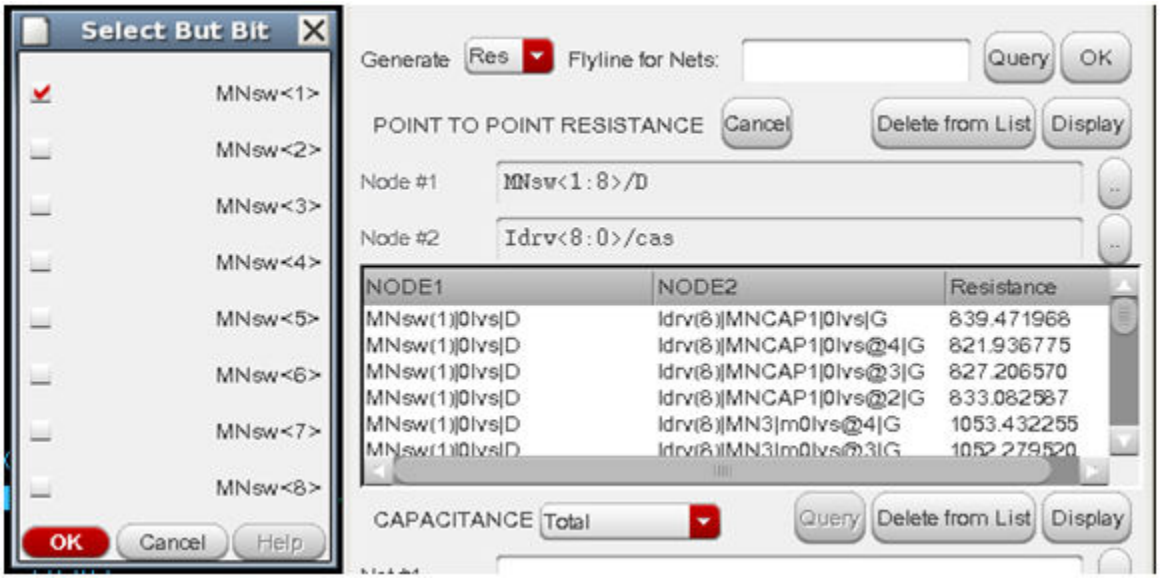


Figure 84 shows an example of probing results when you choose to probe a single bus bit.

Feedback

Figure 84 Probing a Single Bus Bit



Feedback

Displaying Multiple Nets

To display multiple nets, select Display Multiple Nets in the Probe Options dialog box, as shown in Figure 85. Virtuoso Integration displays each net in different color, as shown in Figure 86.

Figure 85 Display Multiple Nets in the Probe Options Dialog Box

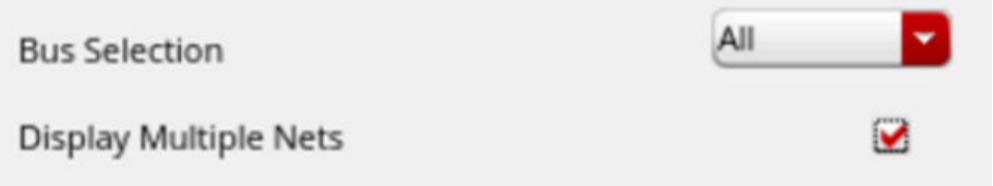
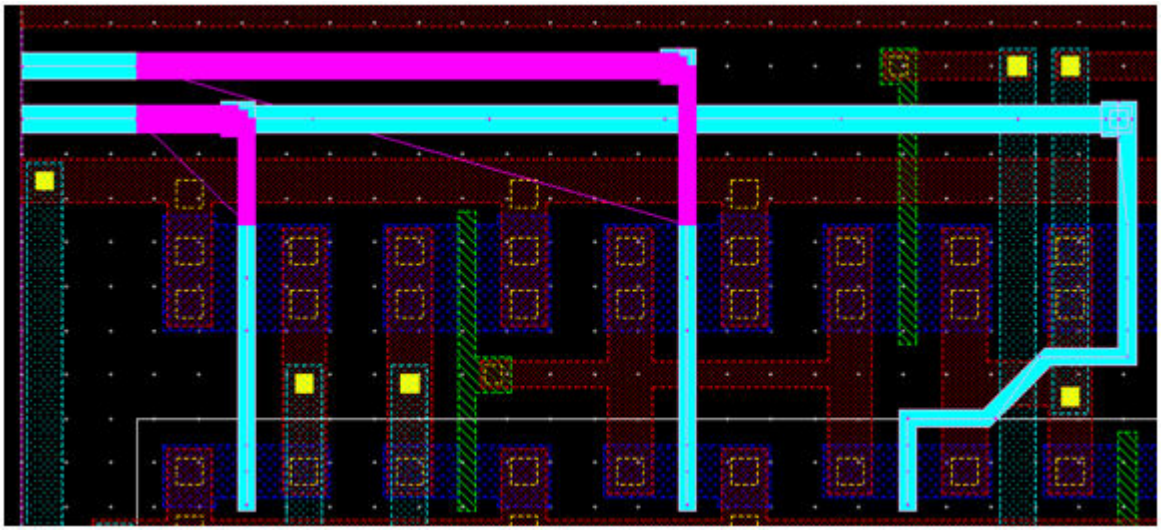


Figure 86 Displaying Multiple Nets



Parasitic View Probing

When you are probing point-to-point resistance inside the parasitic view, it is only possible to query polygons listed as node (that is, port or subnode) layer-purpose pairs in the Runset Layer to DFII Layer mapping file. If no node LPPs were specified in that file, point-to-point parasitic probing is deactivated in the probing utility.

Feedback

Schematic View Probing

Schematic-based probing of parasitic view resistance is available both when probing the top-level schematic corresponding to the parasitic view block and when probing out-of-context by descending the schematic view hierarchy. With this feature, you can probe instance terminals within the schematic view, and corresponding nodes are located within the selected parasitic view. It is possible to probe a hierarchical schematic and achieve resistance results even if the underlying parasitic extraction flattened all hierarchy (that is, `SKIP_CELLS:!*`). If you do not specify the `OBSERVATION_POINTS` command, the StarRC tool adds it to the Virtuoso Integration run.

The matching of instance terminals from a hierarchical schematic to parasitic subnodes from a flattened extraction is accomplished as follows:

- The StarRC `OBSERVATION_POINTS` command generates parasitic subnodes corresponding to hierarchical interactions of cells that are not skip cells. Therefore it is possible to probe the terminal of a cell instance in schematic that does not correspond to a skip cell and still have the parasitic prober find a directly corresponding node in the flat parasitic extraction.
- There are several situations where it is not possible to match a probed schematic instance terminal to a subnode in the parasitic view.

  The `OBSERVATION_POINTS` command only generates nodes when net material in the parent cell interacts with port material in the child cell in layout. If, for example, a top-level net in layout connects through a level-1 instance down to port material inside a level-2 instance, with no port material existing specifically in the level-1 block, no observation point node is generated for the level-0 to level-1 interaction.

  Nodes are only generated for parent- and child-cell interactions when the child cell is listed as a successfully matched EQUIV point (Hercules or IC Validator flow) or HCELL (Calibre flow) in the LVS output.

  When no direct match is found, the parasitic prober searches for all valid parasitic terminal nodes which (a) connect to the same top-level net as does the probed schematic instance terminal, and (b) are located inside the specific instance probed in schematic. All matching parasitic nodes that meet these criteria are used when reporting point-to-point resistance. Therefore, if you probe one schematic terminal that matches M parasitic terminal nodes, and a second schematic terminal that matches N parasitic terminal nodes, a total of M * N point-to-point resistance results are reported.

Probed Results Log and Cross-Probing

As point-to-point resistances are probed within either the schematic view or the parasitic view, results are stored within the results logs (see Figure 74). If a window containing the parasitic view is open, you can select a previously probed resistance entry in the

results log and click Display to highlight or zoom-in to the node shapes between which the resistance value applies. A flyline also appears between the two node shapes. In this manner, you can probe point-to-point resistance results in the schematic view and then select the result in the prober to see it highlighted in the parasitic view.

Note that the highlighting and zooming functions work properly only if node markers are contained inside the parasitic view for the two nodes listed in the selected resistance result. It is possible to probe the schematic and build resistance results in the results log, but be unable to highlight and zoom in to the results due the fact that node shapes were not generated during parasitic view generation. For more information about parasitic view node shapes, see The Layer Mapping File.

Prober File Input and Output

The toolbar of the parasitic prober contains two buttons used to store and load parasitic results between the prober dialog box and a text file.

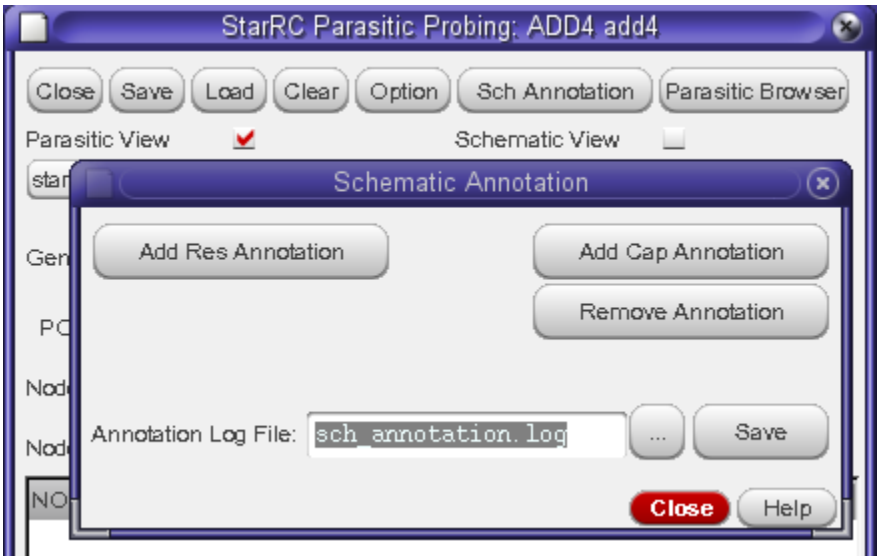
Button	Description
Save to File	Saves all results contained in the resistance results log and capacitance results log to a specified text file.
Load From File	Loads capacitance and resistance results from a specified text file into the prober form results logs.

Schematic Annotation

The Parasitic Prober provides the ability to annotate schematic net names with total capacitance or total resistance information from the parasitic view specified in the prober form. Click the “Sch Annotation” button on the Parasitic Probing dialog box to activate the Schematic Annotation dialog box, as shown in Figure 87.

Feedback

Figure 87 Schematic Annotation Dialog Box



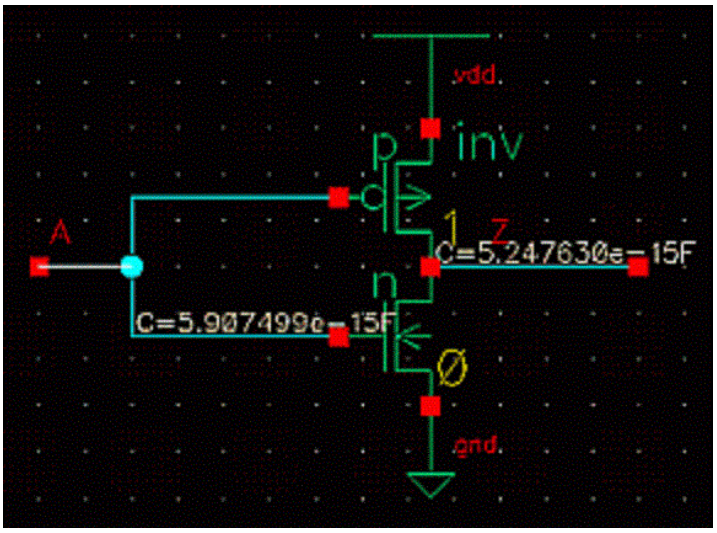
You can save the annotation to a file; the default file name is sch_annotation.log. In the Parasitic Probing dialog box, the Load button allows you to load saved annotation files.

The annotations are highlight labels instantiated on layer-purpose pairs:

```
("annotate" "drawing8")
```

An example of a simple inverter schematic annotated with parasitic capacitance is shown in Figure 88.

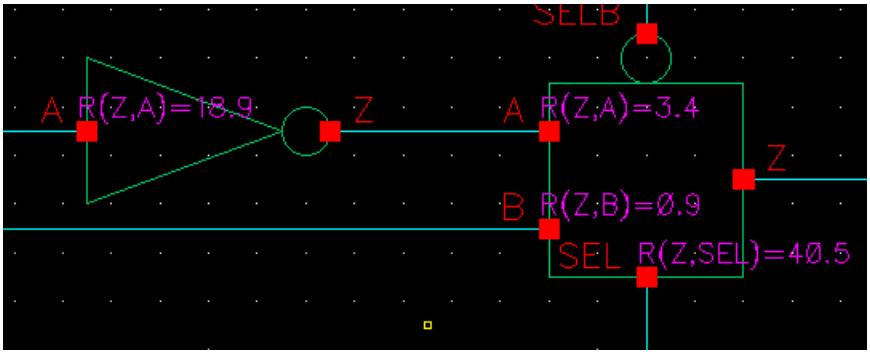
Figure 88 Example of Parasitic Capacitance Annotation



Feedback

An example of parasitic resistance annotation is shown in Figure 89. The label takes the form R(x,y) where x is the driver pin and y is the load pin.

Figure 89 Example of Parasitic Resistance Annotation

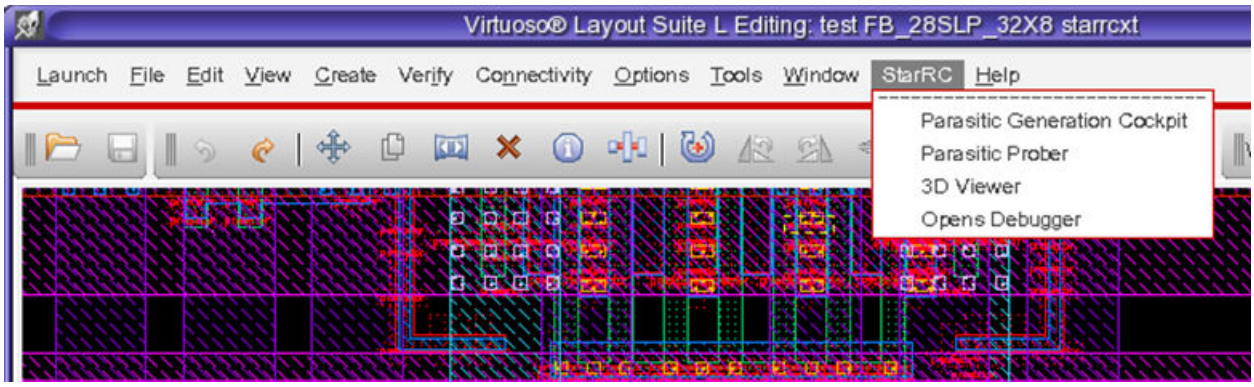


Feedback

Opens Debugger

The StarRC tool provides a utility for debugging opens. Select it in the top-level Virtuoso menu bar, as shown in Figure 90.

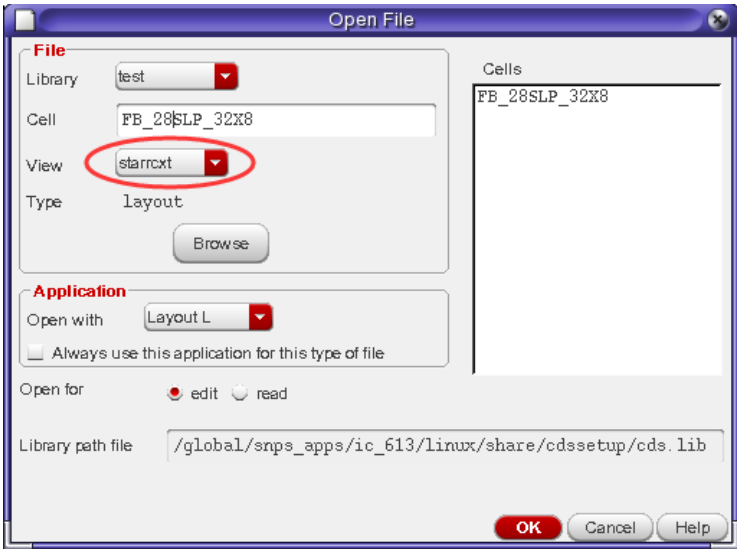
Figure 90 Selecting the Opens Debugger



When setting up the Opens Debugger, select an OA view created by the StarRC tool. For example, the selection in Figure 91 is named starrcxt.

To use the Opens Debugger tool, you must have generated the physical view with the `REDUCTION: NO` and `EXTRA_GEOMETRY_INFO: NODE RES` commands. Otherwise, an error message is issued.

Figure 91 Setting Up the Opens Debugger

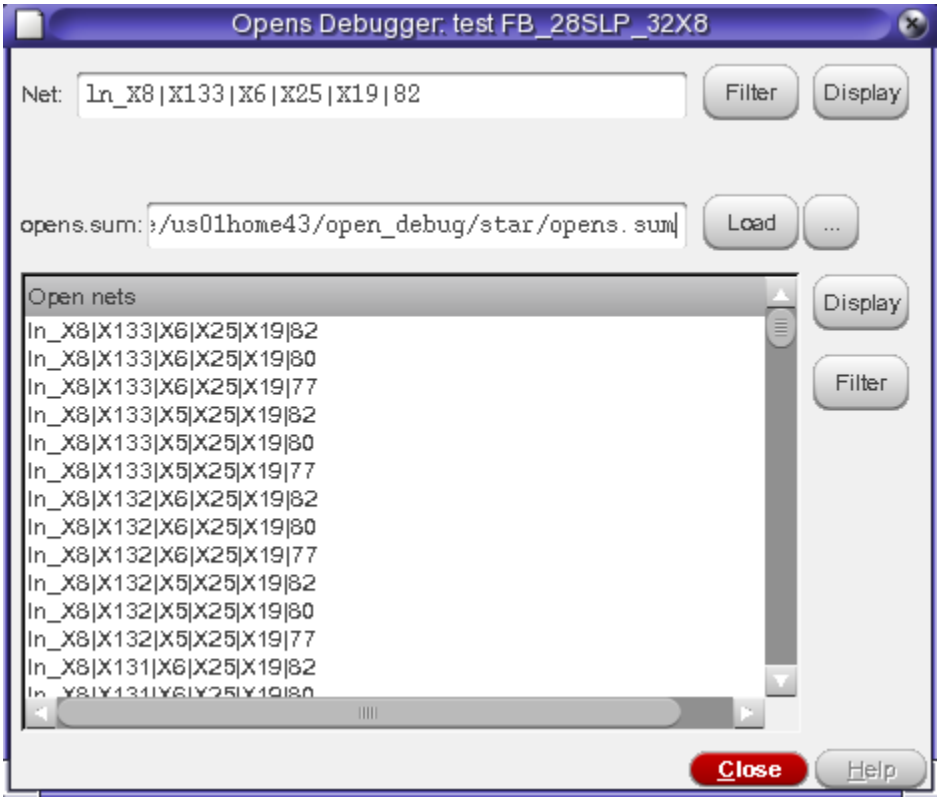


The Opens Debugger dialog box appears, as shown in Figure 92.

To choose a net to display, use one of the following methods:

- Type a net name into the Net field at the top of the form.
- Load an opens.sum file in the middle of the dialog box, then select a net from the resulting list.

Figure 92 Opens Debugger Dialog Box



After specifying a net, use the Filter button to the right of the net name to select Resistively Connected Groups, as shown in Figure 93.

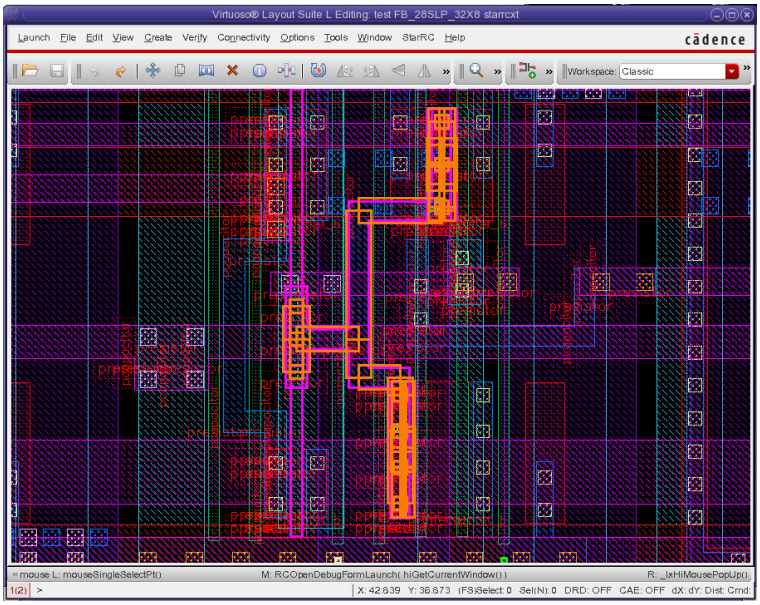
Figure 93 Opens Debugger RCG Selection

A resistively connected group (RCG) is a set of nets that have the same name or ID but have no physical connection. The StarRC tool uses the available layout and schematic information to create related groups of nets to assist the debugging process. Select or deselect the RCG options presented in the window to highlight different groups.

Feedback

Figure 94 shows an example of a highlighted group.

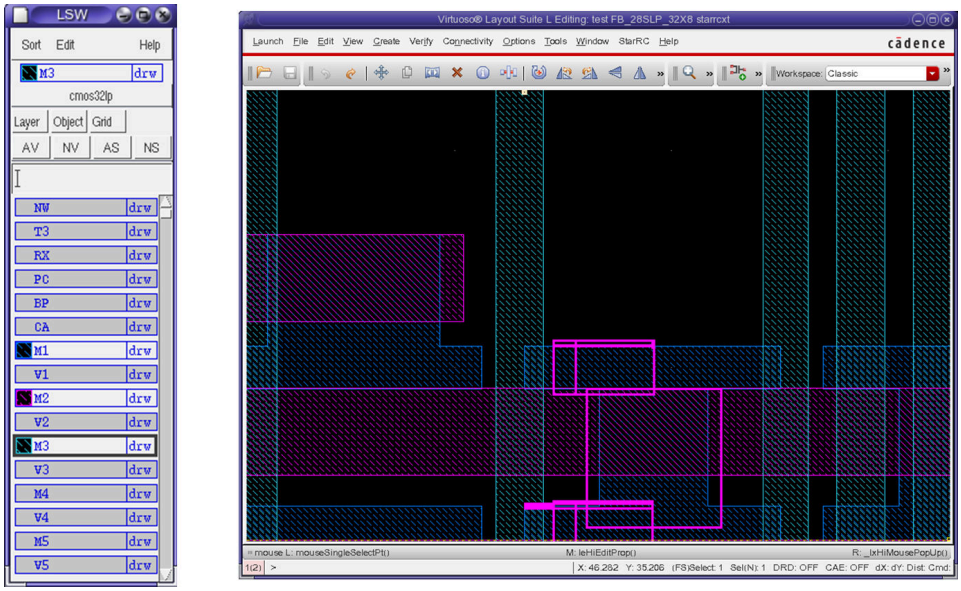
Figure 94 Opens Debugger RCG Display



Feedback

You can choose the layers to display, as shown in Figure 95.

Figure 95 Opens Debugger RCG Display



Feedback

Virtuoso Integration SKILL Procedures and Related Variables

As you load the rcskill.cxt file, you can use two types of SKILL procedures:

- GUI Integration With a Custom Interface
- Batch Mode Procedures

GUI Integration With a Custom Interface

You can customize your user interface with the following features:

- `RC_ADD_MENU` environment variable

  The `RC_ADD_MENU` environment variable controls the StarRC pulldown menu in the layout and schematic windows. This variable must be set before invoking callInitProc to load the StarRC Virtuoso Integration feature; this setting remains in effect for the entire Virtuoso session.

Variable	Definition
`RC_ADD_MENU = YES`	StarRC pulldown menu is shown in the layout and schematic windows
`RC_ADD_MENU = NO`	StarRC pulldown menu not shown in the layout and schematic windows

- `RCProbeFormLaunch` SKILL procedure

  If you set `RC_ADD_MENU=NO` to disable the StarRC pulldown menu, you can use the following SKILL procedure to enable the Parasitic Prober to be launched from your custom user interface:

  ```
  RCProbeFormLaunch(hiGetCurrentWindow())
  ```

- `SetupAllInOneForm` SKILL procedure

  If you set `RC_ADD_MENU=NO` to disable the StarRC pulldown menu, you can use the following SKILL procedure to enable the Cockpit GUI to be launched from your custom user interface:

  ```
  SetupAllInOneForm(hiGetCurrentWindow())
  ```

Feedback

Batch Mode Procedures

You can access batch mode parasitic view generation with the following procedures:

- RCGenParaViewBatch
- RCCockpitRun

RCGenParaViewBatch

The RCGenParaViewBatch procedure accepts only key-based input.

An example of the procedure call is as follows:

```
RCGenParaViewBatch(         ?dfiiInputLib "AD4FUL"
     ?dfiiInputCell "AD4FUL"
     ?dfiiOutputView "starrc"
     ?cmdFile "star_cmd"
     ?dfiiDeviceMap "/design/dfii_devlist.txt"
     ?propmap_case_sensitive "NO"
     ?dfiiLayerMap "/design/dfii_layermap.txt"
     ?extract t
     ?genPhyPolygn t
     ?genFlyline nil
     ?skip_cell_list ""
     ?carry_sch_prop t
     ?sch_view_name "schematic"
     ?lsf_command nil
     ?separate_starrcviewlog nil
     ?vi_bus_bit ""
)
```

This example includes only the minimum necessary set of keys. Other available keys are summarized in Table 43.

A value of `t` for a key means true or yes. A value of `nil` means no.

An alternative method for using the RCGenParaViewBatch procedure is to write a script procedure to invoke it. The enclosing procedure can fix some of the key values while allowing others to be entered on the command line. An example of this method is as follows:

```
procedure(
  RCGenParaViewBatch(
    @key dfiiInputLib dfiiInputCell dfiiOutputView cmdFile
    dfiiDeviceMap dfiiLayerMap extract blockMode runDir
    (subnodeSize "0.1")
    preprocessCallback postprocessCallback
    spiceCardStripInstpathCallback spiceCardStripPrimitiveCallback
    dfiiOutputLib dfiiOutputCell (genPhyPolygn t) (genFlyline nil)
    (skip_cell_list "") (oa_writer t ) (oa_lib_def "") (carry_sch_prop t)
    (sch_view_name "schematic") (lsf_command nil)
```

```
        (propmap_case_sensitive nil)
        (port_annotation_view "")
    )
```

Table 43 Keys Used in the RCGenParaViewBatch Procedure

Key name	Data type	Definition [default, if any]
blockMode	Boolean	Whether to run the StarRC tool in block mode Default: `nil`
carry_sch_prop	Boolean	Pass schematic property or not Default: `t`
cmdFile	string	StarRC command file name
dfiiInputCell	string	Input cell name
dfiiDeviceMap	string	Device mapping file name
dfiiInputLib	string	Input library name
dfiiLayerMap	string	Layer mapping file name
dfiiOutputCell	string	Output cell name
dfiiOutputLib	string	Output lib name
dfiiOutputView	string	Output view name
extract	Boolean	Whether to run extraction Default: `t`
genFlyline	Boolean	Whether to generate flylines Default: `nil`
genPhyPolygn	Boolean	Whether to generate physical polygons Default: `t`
lsf_command	string	Command to submit an LSF job
multi_output	string	Format of parasitic netlist to generate in addition to OA parasitic view. Valid values: `spf`, `star`, `spef` Default: `nil`
oa_lib_def	string	User-specified lib.defs file name Default is "", which uses the Virtuoso lib.defs file
overwrite_locked_view	Boolean	Whether to allow overwriting of locked parasitic views Default: `nil`

Feedback

Table 43 Keys Used in the RCGenParaViewBatch Procedure (Continued)

Key name	Data type	Definition [default, if any]
port_annotation_view	string	View name to annotate pin info
postprocessCallback	string	Callback to call after view generation
preprocessCallback	string	Callback to call before view generation
propmap_case_sensitive	Boolean	Whether the schematic view property annotation is case-sensitive Default: `nil`
runDir	string	StarRC run directory Default: the current working directory
sch_view_name	string	View name to pass schematic property Default: `schematic`
separate_starrcviewlog	Boolean	Whether to generate a separate starrcview.log file for each run Default: `nil`
set_isglobal_to_ground	Boolean	Whether to set isGlobal flag to ground node Default: `t`
skip_cell_list	string	Skip cell mapping file to instantiate skip cells
skip_lowerlevel_inheritedterm	Boolean	Whether to skip lower level inherited term reading Default: `nil`
spiceCardStripInstpathCallback	string	Whether to strip SPICE cards for instance paths. Valid arguments: `t` (do not strip), `nil` (strip), or a string procedural call to a SKILL expression for SPICE-card stripping within the instance path. Default: `nil`
spiceCardStripPrimitiveCallback	string	Whether to strip SPICE cards for primitive devices. Valid arguments: `t` (do not strip), `nil` (strip), a string procedural call to a SKILL expression for SPICE-card stripping of the primitive at the end of the instance path, or a cdlinclude file that names the SUBCKTs for which the primitive X card should be stripped. Default: `nil`
subnodeSize	float	Pin or subnode size in physical view Default: 0.1

Feedback

Table 43 Keys Used in the RCGenParaViewBatch Procedure (Continued)

Key name	Data type	Definition [default, if any]
temperature	string	Temperatures for TemperatureVX flow, separated by spaces
vi_bus_bit	string	Bus bit delimiting characters Default: <>

Some of the keys correspond to StarRC commands, as shown in Table 44.

Table 44 Correspondence to StarRC Commands

Key	StarRC Command
carry_sch_prop	controlled by OA_PROPERTY_ANNOTATION_VIEW
dfiiDeviceMap	OA_DEVICE_MAPPING_FILE
dfiiInputCell	OA_CELL_NAME (if dfiiOutputCell is not set)
dfiiInputLib	OA_LIB_NAME (if dfiiOutputLib is not set)
dfiiLayerMap	OA_LAYER_MAPPING_FILE
dfiiOutputCell	OA_CELL_NAME
dfiiOutputLib	OA_LIB_NAME
dfiiOutputView	OA_VIEW_NAME
genPhyPolygn	controlled by OA_LAYER_MAPPING_FILE
multi_output	OA_MULTI_OUTPUT
oa_lib_def	OA_LIB_DEF
overwrite_locked_view	OA_OVERWRITE_LOCKED_VIEW
port_annotation_view	OA_PORT_ANNOTATION_VIEW
propmap_case_sensitive	OA_PROPMAP_CASE_SENSITIVE
sch_view_name	OA_PROPERTY_ANNOTATION_VIEW
set_isglobal_to_ground	OA_NOT_GLOBAL_NETS
skip_cell_list	OA_SKIPCELL_MAPPING_FILE

Table 44 Correspondence to StarRC Commands (Continued)

Key	StarRC Command
spiceCardStripInstpathCallback nil spiceCardStripInstpathCallback t	OA_REMOVE_SPICECARD_PREFIX YES OA_REMOVE_SPICECARD_PREFIX NO
spiceCardStripPrimitiveCallback nil spiceCardStripPrimitiveCallback T	OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX YES OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX NO
subnodeSize	OA_MARKER_SIZE
vi_bus_bit	OA_BUS_BIT

RCCockpitRun

You can run a StarRC extraction within the Virtuoso environment by loading a batch file with the SKILL interface or by using the StarRC generation cockpit. In either case, a command file named cockpit_cmd is saved in the run directory. You can use this file as the input to an RCCockpitRun batch mode run.

After a successful StarRC run within the Virtuoso environment, several automatically created files remain in the run directory:

- oa_cmd

  A StarRC command file created when the output from the StarRC tool is an OpenAccess format netlist.

- gui_cmd

  A StarRC command file generated from the “additional options” dialog box in the StarRC cockpit.

- view_cmd

  A StarRC command file containing all of the commands necessary for view generation or output netlist generation from the GUI.

- starrcview.log.*time_stamp*

  A file created only during an OpenAccess output extraction run. The file contains a header, the list of StarRC commands used in the run, and the final status (pass or fail). A time stamp with the date and time is automatically appended to the file name.

- viewlog.*block_name*.txt

  A log file showing commands and status. Including the block name in the file name is optional but might be useful if you are running multiple extractions. To include the block name, select the check box labeled Separate viewlog.txt By Block” in the “Run Cockpit” tab. If you do not select this option, the log file is named viewlog.txt.

You can also create an optional command file to perform additional actions. The following is the command that you would execute from the unix command line, where user_cmd is the user-created command file:

```
% StarXtract gui_cmd user_cmd view_cmd oa_cmd
```

The order in which the files appear on the command line is important. In this example, the user-created command file overrides the gui_cmd file, but the view_cmd and oa_cmd files override the user-created command file.

An RCCockpitRun Virtuoso Integration job can be rerun if the cockpit_cmd, gui_cmd, and view_cmd files are kept. To replicate a StarRC run within the Virtuoso environment, enter `RCCockpitRun("cockpit_cmd")` in the Cadence CIW window.

9

Transistor-Level Extraction

This chapter contains information about transistor-level extraction and instructions for modifying an IC Validator runset or Calibre rule file for use with the StarRC tool.

For more information, see the following topics:

Feedback

Preparing IC Validator Runsets

The following section explains the rules and the required functions in the IC Validator runsets.

Runset Rules

When creating an IC Validator runset, you must follow certain rules in data manipulation.

The following layer definitions are accepted by IC Validator:

- Runset layer – Any layer specified in the connect section of the IC Validator runset
- Physical layer – Any layer specified as a `CONDUCTOR` or `VIA` in the ITF file

This section describes the following rules:

- A runset layer via can connect only two physical layers. See Via Layer Rule.
- All device runset layers should be negated with the routing runset layers. See Device Layer Rule.
- A physical layer cannot directly connect to another physical layer without using a via unless the two physical layers are covertical. See Physical Layer Rule.

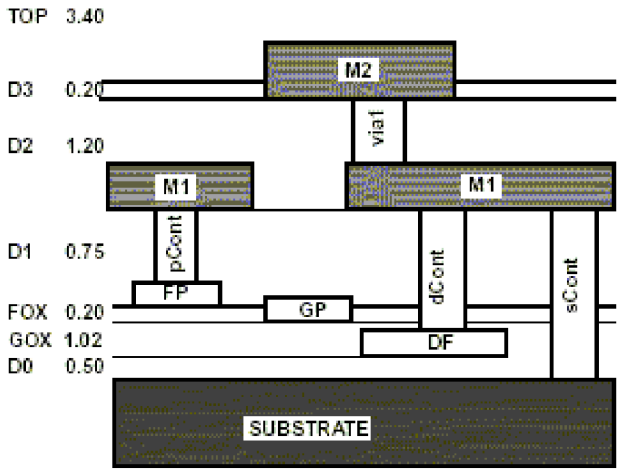
Via Layer Rule

```
input_cdb = connect(
   connect_items = {
        {{M1, poly}, cont},
        {{M1, nsd, psd}, cont}
….
};
```

Separating metal1 to diffusion contacts (dCont) and metal1 to poly contacts (pCont) is necessary because these two contacts connect metal1 to two different physical layers at different levels in the ITF file. In SOI or STI processes where diffusion and substrate exist on two different physical levels in the ITF file, distinguishing the metal1 to substrate contacts (sCont) might also be necessary. Metal1 to substrate contacts, where diffusion and substrate exist on separate physical levels in the ITF file, are shown in Figure 96.

Feedback

Figure 96 Separate Metal1 to Diffusion Contacts



For most runsets, following the correct convention is straightforward. In general, the runset should have specific contacts for connecting the following physical layers:

```
…
metal2 - metal1
metal1 - poly
metal1 - diffusion and SUBSTRATE

CORRECT:
polyCont = cont and poly;
subCont = cont not polyCont;
input_cdb = connect(
   connect_items = {
        {{m1, poly}, polyCont},
        {{m1, nsd, psd}, subCont}
…
};
```

Device Layer Rule

Removing portions of routing layers that overlap device layers allows the StarRC tool to ignore device capacitances correctly, and is necessary for MOS gate, source, and drain terminals as well as for capacitance terminals.

Incorrect:

```
ngate = poly and ndiff;
nsd = ndiff not poly;
….

nmos(my_devices, "n", nsd, ngate, nsd, {{psub}});
….
```

Feedback

```
cap_top = m2 and cap_recog;
cap_bot = m1 not cap_bot;
cap_dev = cap_top and cap_bot;
….
capacitor(my_devices, "m1m2cap", cap_dev, cap_top, cap_term,
area_capval=1e-15);
……
input_cdb = connect(
   connect_items = {
        {{poly}, ngate},
        {{cap_top}, m2},
        {{cap_bot}, m1},
…
};
```

The StarRC tool does not ignore the gate capacitance correctly and the designed capacitance is double-counted in this example. Other devices might not have these issues.

Correct:

```
ngate = poly and ndiff;
nsd = ndiff not poly;
poly = poly not ngate;
my_devices = init_device_matrix(netlist_cdb);
nmos(my_devices, "n", nsd, ngate, nsd, {{SUBSTRATE}});
cap_top = m2 and cap_recog;
cap_bot = m1 and cap_recog;
m2 = m2 not cap_top;
m1 = m1 not cap_bot;
capacitor(my_devices, "m1m2cap", cap_dev, cap_top, cap_bot,
area_capval=1e-15);
input_cdb = connect(
  connect_items = {
    {{poly}, ngate},
    {{cap_top}, m2},
    {{cap_bot}, m1},
    …
    }
);
```

The `not()` function performed on the gate and field poly is required for the planar and nonplanar processes because the `IGNORE_CAPACITANCE` capability behaves the same in both cases. A planar process implies that the gate poly and the field poly layers are both mapped to a single POLY layer, while a nonplanar process implies that the gate poly and the field poly are mapped to separate covertical layers in the nxtgrd file.

Physical Layer Rule

In Figure 96, metal1 connects to metal2 by via1. However, placing a contact between physical layers FP and GP is not necessary because they overlap on the vertical profile.

Feedback

These two conductors are therefore covertical. In the previous runset example, connecting poly and ngate layers is allowed because the two runset layers map to the covertical physical layers FP and GP. Likewise, connecting cap_top to m2 is allowed because both are mapped to the same physical layer M2. The same also applies for cap_bot and m1.

Required Functions

Add the following required functions to the IC Validator runset script to ensure that extraction runs correctly:

```
pex_generate_results(
    pex_matrix = pex_matrix,
    device_extraction_matrix = my_devices,
    device_db = device_db,
    layout_database = mw_handle,
    pex_process_map_file = pex_process_handle,
    pex_runset_report_file = pex_report_handle
);
```

In addition, a tag name can be used to assign a layer name to runset layers. In Example 11 “SUBSTRATE” and “via2” are tag names shown in the runset report file and extract view layer name.

Example 11 Tag Name Use in IC Validator Run Script

```
pex_conducting_layer_map(pex_matrix, psub, "SUBSTRATE");
pex_via_layer_map(pex_matrix, V2, "via2");
```

The StarRC tool uses the IC Validator runset report file to obtain LVS results and perform parasitic extraction. The key argument is `pex_runset_report_file` in the `pex_generate_result()` function. The default for `pex_runset_report_file` is “`./pex_runset_report`”.

After an IC Validator run is completed, the XTR view is written into the LIB_NAME directory under the default directory path, $working_directory/XTROUT.

To change the defaults for variables related to IC Validator, examine the $ICV_HOME_DIR/include/rcxt_public.rh file.

The content of this file is similar to the following example:

```
pex_generate_results : published function (
    pex_matrix                : pex_layer_matrix,
    device_extraction_matrix  : device_matrix,
    device_db                 : device_database,
    layout_database           : in_out milkyway_library_handle,
    pex_process_map_file      : pex_process_map_file_handle,
    pex_runset_report_file    : pex_runset_report_file_handle,
    pex_lpp_map_file          : pex_lpp_map_file_handle =
NULL_LPP_MAP_FILE,
```

Feedback

```
    pex_tagname_required     : boolean = true,
    precision                : integer = 3
) returning void;
```

Hierarchy Options

To specify hierarchical options, use the `hierarchy_options()` function in IC Validator. This is the same as the `technology_options` function in Hercules. The `hierarchy_options()` option controls the hierarchy optimization.

Note:

When you use the `hierarchy_options` option, some of the vcell options create new cells that do not exist in the schematic and hinder the `XREF:YES` or `XREF:COMPLETE` flows. In this case, set the `iterate_max` to `0` in both pairs and sets of stages.

Hierarchy Options Syntax

The syntax used to specify hierarchy options is shown in the following example:

```
hierarchy_options (
        pairs = {{pair_cells = {"string", …},
           iterate_max = integer,
           explode_into_vcell = ON | OFF | AUTO,
           x_pairing = true | false,
           y_pairing = true | false},
           … }, //optional
        sets = {{base_cells = {"string", …},
        programming_cells = {"string", …},
        iterate_max = integer,
        explode_into_vcell = ON | OFF | AUTO,
        flatten_sets = true | false,
        min_cell_overlap = integer,
        share_base_cells = true | false},
        …}, //optional
 );
```

Runset Example

The following example is a typical IC Validator runset for a transistor-level extraction.

```
#include "primeyield.rh"

library(
    format = GDSII,
    cell = "add4",
    library_name = "ADD4.GDS"
);
```

Feedback

```
schematic_netlist_db = schematic(
    schematic_file    = {{"ADD4.sp",SPICE}}
);

#include "equiv.rs"

NDIFF = assign({{1}});
PDIFF = assign({{2}});
NWELL = assign({{3}});
POLY  = assign({{5}});
POLY_TEXT = assign_text({{25}});
CONT  = assign({{6}});
M1    = assign({{8}});
M1_TEXT = assign_text({{28}});
V1    = assign({{9}});
M2    = assign({{10}});
M2_TEXT = assign_text({{30}});
V2    = assign({{44}});
M3    = assign({{45}});
M3_TEXT = assign_text({{32}});

sub1 = cell_extent(
    cell_list = {"*"}
);

psub = sub1 not NWELL;
pactive = PDIFF and NWELL;
nactive = NDIFF not NWELL;
subtie  = PDIFF not NWELL;
welltie = NDIFF and NWELL;
pactive = PDIFF not subtie;
nactive = NDIFF not welltie;
ngate = POLY and nactive;
pgate = POLY and pactive;
fpoly = POLY not (ngate or pgate);
nsd   = nactive not ngate;
psd   = pactive not pgate;

input_cdb = connect(
   connect_items = {
        {{M3, M2}, V2},
        {{M2, M1}, V1},
        {{M1, fpoly}, poly_con},
        {{ngate, pgate}, fpoly},
        {{M1, nsd, psd}, diff_con},
        {{M1, welltie, subtie}, diff_con},
        {{NWELL}, welltie},
        {{psub}, subtie}
   }
);

netlist_cdb = text_net(
   connect_sequence = input_cdb,
```

Feedback

```
   text_layer_items = {
      { M3, M3_TEXT },
      { M2, M2_TEXT },
      { M1, M1_TEXT },
      { fpoly, POLY_TEXT }
   },
   attach_text = ALL
);

my_devices = init_device_matrix(netlist_cdb);

nmos(my_devices, "n", nsd, ngate, nsd, {{psub}});
pmos(my_devices, "p", psd, pgate, psd, {{NWELL}});

device_db = extract_devices(my_devices);

layout_netlist_db = netlist(device_db);

compare_settings = init_compare_matrix();

compare(compare_settings, schematic_netlist_db, layout_netlist_db,
 generate_equiv = {FULL_NAME}, write_equiv_netlists=ALL);

pex_matrix = init_pex_layer_matrix(my_devices);
pex_conducting_layer_map(pex_matrix, M3, "metal3");
pex_conducting_layer_map(pex_matrix, M2, "metal2");
pex_conducting_layer_map(pex_matrix, M1, "metal1");
pex_conducting_layer_map(pex_matrix, fpoly, "poly");
pex_conducting_layer_map(pex_matrix, ngate, "poly");
pex_conducting_layer_map(pex_matrix, pgate, "poly");
pex_conducting_layer_map(pex_matrix, nsd, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, psd, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, welltie, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, subtie, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, NWELL, "SUBSTRATE");
pex_conducting_layer_map(pex_matrix, psub, "SUBSTRATE");

pex_via_layer_map(pex_matrix, V2, "via2");
pex_via_layer_map(pex_matrix, V1, "via1");
pex_via_layer_map(pex_matrix, poly_con, "polyCont");
pex_via_layer_map(pex_matrix, diff_con, "diffCont");

pex_process_handle = pex_process_map_file();
pex_report_handle = pex_runset_report_file();
mw_handle = milkyway_library("XTROUT");

pex_generate_results(
    pex_matrix = pex_matrix,
    device_extraction_matrix = my_devices,
    device_db = device_db,
    layout_database = mw_handle,
    pex_process_map_file = pex_process_handle,
```

Feedback

```
    pex_runset_report_file = pex_report_handle
);
```

IC Validator Marker Generation

The following sections describe text and layer markers in the IC Validator tool.

Text-Based Marker Layers

The `text_origin()` function is used for the IC Validator flow. It generates a shape surrounding the selected text in the layout. The shape should be very small, because it is included in the CONNECT sequence and can create shorts.

In the IC Validator runset (runset.rs):

```
metal1          = assign({{6}});
metal1_text     = assign_text({{20}, {30}});
```

In the previous example, markers = text_origin(metal1_text, shape_size=0.01, cells = {"XT_DEVICES"}, text = {"*", "VDD", "VSS"}); m1_markers = markers interacting metal1.

```
input_cdb = connect(
   connect_items = {
            …..
            {{metal1}, m1_markers},
       ….
   }
);
netlist_cdb = text_net(
   connect_sequence = input_cdb,
   text_layer_items = {
      { metal1, metal1_text },
      ….
   },
   attach_text = ALL
);
…
```

In the StarRC mapping file:

```
conducting_layers
metal1
…
marker_layers
m1_markers
```

Example of ID-Layer Markers

The simplest approach to manual marker generation, this technique uses a pre-existing input layer (assign()) to identify marker shapes.

Feedback

The following is defined In the IC Validator runset:

```
metal1          = assign({{6}});
metal1_text     = assign_text({{20}, {30}});
…
metal1_pio   = metal1 and PAD
metal1_markers = metal1_pio or metal1_pin
input_cdb = connect(
   connect_items = {
        …
        {{metal1}, m1_markers},
       …

   }
);
netlist_cdb = text_net(
   connect_sequence = input_cdb,
   text_layer_items = {
      { metal1, metal1_text },
   },
   attach_text = ALL
);
```

In the StarRC mapping file:

```
conducting_layers
metal1
…
marker_layers
metal1_markers
```

Feedback

Multifinger Device Support

Multifinger devices have multiple gate, source, and drain terminals. However, in the XTR view of the Milkyway database, a device can only have a single gate, source, or drain terminal. The XTR view writer in the IC Validator tool stores the extra terminal information inside the terminal properties, which is parsed by the StarRC tool.

When a device function is declared in the IC Validator tool with the option settings `recognition_layer=true` and `merge_parallel=true`, the individual coordinates of each polygon in terminals with multiple polygons are reported.

The StarRC tool reads each set of coordinates from the XTR view for correct resistance network calculation.

Preparing Calibre Rule Files

The following topics describe the preparation of a rule file for the Calibre Connectivity Interface flow:

- Rule File Creation
- Required Commands
- Support for Calibre Preprocessor Directives and Include Statements
- Rule File Example

Rule File Creation

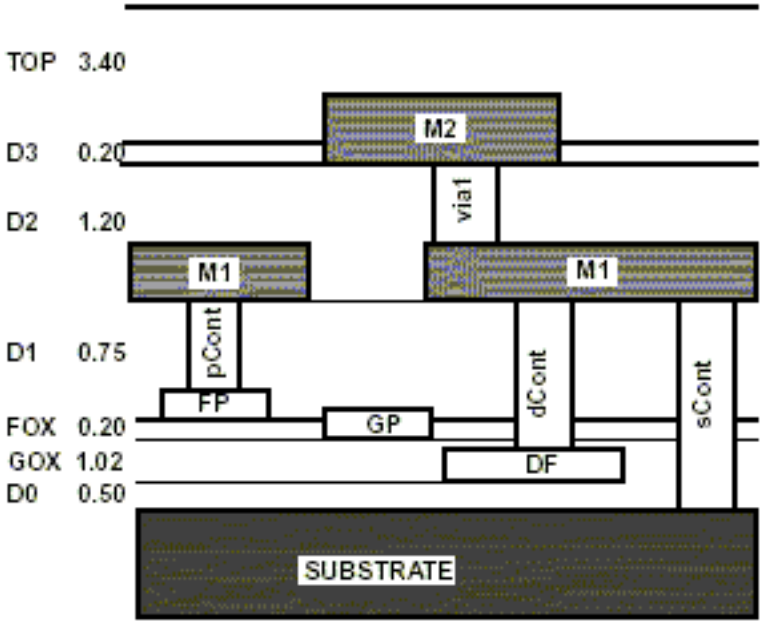
Use the following guidelines when creating a Calibre runset:

- A runset layer can connect only two physical layers.

  In the Calibre rule file, it is important to separate the metal1 to diffusion contact from the metal1 to poly contact. LVS conventions typically allow both contacts to exist on the same rule file layer (“cont” in this example). However, physically these are separate contacts, so it is important for them to be separated for the StarRC extraction engine. See Figure 97. Moreover, these contacts also have different resistivity.

Feedback

Figure 97 Separate Calibre Runset Layers



Incorrect:

```
CONNECT m1 poly BY cont
CONNECT m1 nsd psd BY cont
```

Correct:

```
polyCont = cont AND poly
subCont  = cont NOT poly

CONNECT m1 poly BY polyCont
CONNECT m1 nsd psd BY subCont
```

- Device layers overlap with routing layers should be eliminated by using the NOT operation.

  It is important to remove from routing layers any overlap that exists with device terminal layers. The StarRC `IGNORE_CAPACITANCE` functionality is based on device terminal layers. If there are locations where routing layers overlap terminal layers where the two are mapped to the same physical layer, the StarRC tool ignores capacitance to the terminal layer but retains capacitance to the routing layer. Thus, any such overlap must be removed from the routing layer to prevent the inadvertent inclusion of MOS device capacitances when such capacitances are to be ignored, and to prevent the inclusion of parasitic capacitance between the terminals of a designed capacitor device.

Incorrect:

```
ngate = poly AND ndiff
nsd = ndiff NOT poly
DEVICE MN(n) ngate ngate (G) nsd (S) nsd (D) psub (B)

cap_top = m2 AND cap_recog
cap_bot = m1 AND cap_recog
cap_dev = cap_top AND cap_bot
DEVICE C(m1m2cap) cap_dev cap_top (POS) cap_bot (NEG)

CONNECT poly ngate
CONNECT cap_top m2
CONNECT cap_bot m1
```

Correct:

```
ngate = poly AND ndiff
nsd   = ndiff NOT poly
poly_route = poly NOT ngate
DEVICE MN(n) ngate ngate (G) nsd (S) nsd (D) psub (B)

cap_top = m2 AND cap_recog
cap_bot = m1 AND cap_recog
cap_dev  = cap_top AND cap_bot
m2_route = m2 NOT cap_top
m1_route = m1 NOT cap_bot
DEVICE C(m1m2cap) cap_dev cap_top (POS) cap_bot (NEG)

CONNECT poly_route ngate
CONNECT cap_top m2_route
CONNECT cap_bot m1_route
```

- A physical layer cannot connect to another physical layer without using a via, unless the two physical layers are covertical.

  In the rule file example, the poly_route and ngate layers are allowed to directly connect without a via because the two runset layers map either to the same physical layer (if field and gate poly are not separated in the ITF file) or to covertical physical layers (if field and gate poly are separated in the ITF file). Similarly, the cap_top layer is allowed to directly connect to the m2_route layer because both are mapped to the same physical layer M2. The same is true for the cap_bot and m1_route layers.

- Simple STAMP commands in the Calibre rule file are supported. However, the StarRC tool ignores composite STAMP commands and issues an SX-1242 warning message.

Feedback

Required Commands

The following commands are required in the Calibre rule file to ensure the Calibre Connectivity Interface query server can properly generate StarRC input files:

- `MASK SVDB DIRECTORY "svdb" CCI`

  The Calibre Connectivity Interface flag ensures that the Calibre tool writes all required information to the svdb directory such that the Calibre query server can generate all Calibre Connectivity Interface outputs required for StarRC extraction.

- `PORT LAYER TEXT`

  Top-level ports are identified by text layers designated as PORT LAYER TEXT in the Calibre rule file. This information in turn is used by the Calibre Connectivity Interface query server to generate the CALIBRE_CELLS_PORTS file, which the StarRC tool uses to netlist top-level ports (*|P or *P) in the parasitic netlist. Thus, when top-level ports are required in the parasitic netlist, it is essential that PORT LAYER TEXT be used in the Calibre rule file.

Support for Calibre Preprocessor Directives and Include Statements

The StarRC tool can correctly read the Calibre rule file preprocessor directives #DEFINE, #UNDEFINE, #IFDEF, #IFNDEF, and #ENDIF. In interpreting #IFDEF and #IFNDEF statements, the tol requires that conditional names be defined with #DEFINE directives directly within the rule file.

The StarRC tool does not support names defined outside the rule file as shell environment variables. Such names are typically prefixed by a dollar sign ($) within #IFDEF and #IFNDEF directives. If the StarRC tool encounters a conditional within the rule file that uses a shell variable prefixed by $, and that $-prefixed variable does not occur previously in the rule file within an explicit #DEFINE directive, the tool interprets the variable as undefined and issues the following warning:

```
WARNING: StarXtract
WARNING: CALIBRE_RUNSET preprocessor directives which refer
WARNING: to shell or ENV variables are not supported.
WARNING: Conditional expressions which refer to external
WARNING: variables will evaluate as though the variable is undefined.
WARNING: See layers.sum for a full list of affected directives.
```

The StarRC tool also reads Calibre rule file INCLUDE statements that allow one rule file to instantiate another rule file. In cases where preprocessor directives occur across rule files instantiated with INCLUDE, the StarRC tool correctly interprets the directives and applies them to all conditionals across the rule file hierarchy.

Feedback

Rule File Example

The following is a typical Calibre Connectivity Interface rule file for StarRC transistor-level extraction.

```
LAYOUT SYSTEM GDSII
LAYOUT PATH "XT_DEVICES.GDS"
LAYOUT PRIMARY "XT_DEVICES"
LAYOUT ERROR ON INPUT YES

LVS POWER NAME VDD
LVS GROUND NAME VSS
MASK SVDB DIRECTORY "svdb" CCI
TEXT DEPTH PRIMARY
PRECISION 100

// GDSII Input Layer Mapping
LAYER NWELL   1
LAYER ACTIVE  2
LAYER POLY    3
LAYER BORON   4
LAYER CONTACT 5
LAYER METAL1  6
LAYER VIA     7
LAYER METAL2  8
LAYER PWELL   10
LAYER CAP_REG 12
LAYER POLYRES 40
LAYER BJT_REG 13
LAYER DIODE   14

// GDSII Input Text Layer Mapping
TEXT LAYER 20
LAYER MAP 20  TEXTTYPE >= 0  20
TEXT LAYER 30
LAYER MAP 30  TEXTTYPE >= 0  30

// Layer Derivations
psub1           = EXTENT
diode_active    = INTERACT ACTIVE DIODE
bjt_active      = INTERACT ACTIVE BJT_REG
active1         = (ACTIVE NOT diode_active) NOT bjt_active
bjt_nwell       = INTERACT NWELL BJT_REG
nwell1          = NWELL NOT bjt_nwell
psub            = psub1 NOT nwell1
poly_cont       = POLY AND CONTACT
diff_cont       = CONTACT NOT poly_cont
res_poly        = INTERACT POLY POLYRES
poly1           = POLY NOT res_poly
res_body        = res_poly AND POLYRES
res_term        = res_poly NOT res_body
pactive         = BORON AND active1
```

Feedback

```
nactive          = active1 NOT pactive
diff             = pactive OR nactive
pgate            = poly1 AND pactive
5tngate          = INTERACT (poly1 AND nactive) PWELL
ngate            = NOT INTERACT (poly1 AND nactive) PWELL
psd1             = pactive NOT pgate
nsd1             = (nactive NOT ngate) NOT 5tngate
subtap           = psd1 NOT nwell1
weltap           = (nsd1 AND nwell1) NOT PWELL
psd              = psd1 NOT subtap
pplus            = diode_active AND BORON
nplus            = diode_active NOT BORON
emitter          = BORON AND bjt_active
bjt_nwell_ntap   = active1 AND bjt_nwell
5tnsd            = INTERACT nsd1 5tngate
nsd              = (nsd1 NOT weltap) NOT 5tnsd
poly_cap_term    = (poly1 AND METAL1) AND CAP_REG
fpoly            = (poly1 NOT poly_cap_term) NOT diff
metal1_cap_term  = (METAL1 AND poly1) AND CAP_REG
m1               = METAL1 NOT metal1_cap_term
dpn_nwell_term   = INTERACT NWELL pplus
dnp_psub_term    = INTERACT psub nplus

// Text Attachments
ATTACH 20 m1
PORT LAYER TEXT 20
ATTACH 30 METAL2
PORT LAYER TEXT 30

// Layer Connectivity
CONNECT m1 metal1_cap_term
CONNECT METAL2 m1 metal1_cap_term BY VIA
CONNECT m1 psd nsd weltap subtap 5tnsd BY diff_cont
CONNECT m1 nplus pplus emitter bjt_nwell_ntap BY diff_cont
CONNECT metal1_cap_term psd nsd weltap subtap 5tnsd         BY
diff_cont
CONNECT metal1_cap_term nplus pplus emitter bjt_nwell_ntap
BY diff_cont
CONNECT m1 fpoly res_term poly_cap_term                     BY
poly_cont
CONNECT metal1_cap_term fpoly res_term poly_cap_term        BY
poly_cont
CONNECT poly_cap_term fpoly
CONNECT pgate ngate 5tngate fpoly
CONNECT NWELL dpn_nwell_term weltap
CONNECT psub dnp_psub_term subtap PWELL
CONNECT bjt_nwell bjt_nwell_ntap

// Device Definitions
DEVICE D(DPN) pplus pplus (POS) dpn_nwell_term (NEG)
DEVICE D(DNP) nplus dnp_psub_term (POS) nplus (NEG)
DEVICE MP(p) pgate pgate (G) psd (S) psd (D) NWELL (B)<diff>
```

Feedback

```
[
   property L, W, AD, PD, AS, PS
   W = ( perimeter_coincide(pgate,psd) / 2)
   A = AREA( pgate )
   L = A/W
   AD = area(D) * W / perimeter_inside(D, diff)
   AS = area(S) * W / perimeter_inside(S, diff)
   PD = perimeter(D) * W / perimeter_inside(D, diff)
   PS = perimeter(S) * W / perimeter_inside(S, diff)
]
DEVICE MN(n) ngate ngate (G) nsd (S) nsd (D) psub (B) <diff>
[
   property L, W, AD, PD, AS, PS
   W = ( perimeter_coincide(ngate,nsd) / 2)
   A = AREA( ngate )
   L = A/W
   AD = area(D) * W / perimeter_inside(D, diff)
   AS = area(S) * W / perimeter_inside(S, diff)
   PD = perimeter(D) * W / perimeter_inside(D, diff)
   PS = perimeter(S) * W / perimeter_inside(S, diff)
]

DEVICE MN(5tn) 5tngate 5tngate (G) 5tnsd (S) 5tnsd (D)
PWELL (B) NWELL (B2) <diff>
[
   property L, W, AD, PD, AS, PS
   W = ( perimeter_coincide(5tngate,5tnsd) / 2)
   A = AREA( 5tngate )
   L = A/W
   AD = area(D) * W / perimeter_inside(D, diff)
   AS = area(S) * W / perimeter_inside(S, diff)
   PD = perimeter(D) * W / perimeter_inside(D, diff)
   PS = perimeter(S) * W / perimeter_inside(S, diff)
]

DEVICE R(poly_res) res_body res_term (POS) res_term (NEG)
DEVICE C(poly_cap) CAP_REG poly_cap_term (POS)
metal1_cap_term (NEG)
DEVICE Q(pbjt) psub psub (C) bjt_nwell (B) emitter (E)

// COMPARE section
SOURCE SYSTEM SPICE
SOURCE CASE YES
SOURCE PATH "XT_DEVICES.sp"
SOURCE PRIMARY "XT_DEVICES"
LAYOUT CASE YES

LVS REPORT report.lvs
LVS REPORT OPTION A B C D
LVS REPORT MAXIMUM 100
LVS IGNORE PORTS NO
LVS REDUCE SPLIT GATES YES
LVS RECOGNIZE GATES ALL
```

Feedback

```
LVS COMPARE CASE NAMES TYPES
LVS CHECK PORT NAMES NO
LVS REDUCE PARALLEL MOS YES
LVS NL PIN LOCATIONS YES
LVS COMPONENT SUBTYPE PROPERTY mode
```

Preparing the Mapping File

This information explains the rules for preparing mapping files for use with the Hercules, IC Validator, and Calibre tools. A mapping file example is included.

Mapping Rules

The mapping file maps the layers in CONNECT sequences to physical layers in the ITF file.

No ideal layer Is allowed

Every layer in the CONNECT section of the Hercules or IC Validator runset or Calibre rule file needs to be mapped to a conductor or via in the ITF file. Similarly, every device terminal layer needs to be mapped to an ITF conductor layer to ensure devices are properly connected to the parasitic mesh in the parasitic netlist. Nonterminal device recognition layers should not be mapped, with one exception: RES device body layers should be mapped either as conducting layers (to include the impact of the RES body on surrounding nets) or remove_layers (to ignore the impact or the RES body on surrounding nets).

If a runset layer CONNECT or device terminal layer cannot be mapped to any physical layer, it should be listed under the remove_layers section of the mapping file.

Bulk layer mapping must be considered

A bulk layer is any layer that serves as the bulk terminal connection in any device extraction command. Bulk layers mapped in the `conducting_layers` section of the mapping file are used during parasitic extraction only if the `TRANSLATE_RETAIN_BULK_LAYERS` command is used with options other than the default.

The mapping of bulk and nonbulk layers is advisable under the following circumstances:

- Power net extractions where capacitance to well layers should be generated in the netlist as coupling capacitance
- Extractions containing well devices (for example, well resistors, diodes, BJTs) whose database terminal layers consist solely of bulk layers
- Scenarios in which the bulk terminals of designed devices should be generated in the netlist as instance port subnodes instead of ideal nodes

The StarRC tool makes a distinction between true bulk layers and other layers that are physically part of the substrate, such as wells. These are referred to as nonbulk layers.

If there are any other layers in any CONNECT sequences that connect solely to bulk layers, these layers should be mapped in the same manner as the bulk layers to which they connect. A runset layer can be designated as a conducting layer only if the layer has connectivity to other translated runset layers in the design.

Map port layers as marker layers to obtain top-level ports

Top-level ports in the Hercules and IC Validator flows are identified by the existence of polygons mapped as marker layers. In `MARKER_GENERATION: AUTOMATIC` mode, such layers are generated by the use of `TEXT_POLYGON` commands in the Hercules runset.

No such mapping is required in the Calibre Connectivity Interface flow, because top-level markers are identified by listings in the CALIBRE_CELLS_PORTS file output by the Calibre Connectivity Interface query server.

Mapping File Example

In the following mapping file example, the asterisk (`*`) indicates comments.

```
conducting_layers
   * Routing layers
   Metal2           metal2
   Metal1           metal2
   field_poly       poly
   Nwell            SUBSTRATE
   welltie          SUBSTRATE
   subtie           SUBSTRATE
   * Device layers connected to routing layers
   ngate            gate
   pgate            gate
   Cap1             metal1
   Cap2             poly
   pres_body        poly
   * Device layers built into the SUBSTRATE
   nsd              SUBSTRATE
   psd              SUBSTRATE
   PnAnodeTerm      SUBSTRATE
   PnCathodeTerm    SUBSTRATE
   NpnEmitter       SUBSTRATE
   NpnCollector     SUBSTRATE
   NpnBase          SUBSTRATE

via_layers
   * Via layers
   Via1             via1
   PolyCont         poly_con
   SubCont          sub_con
```

```
    NpnEmitterCont   sub_con
    NpnCollectorCont sub_con
    NpnBaseCont      sub_con

marker_layers (only required in Hercules flow)
    * Marker layer
    Metal2_Mark

remove_layers
```

Feedback

Running the Calibre Query Server

This section describes the Calibre query server commands required to generate proper Calibre Connectivity Interface files. The commands listed in the query server command file shown in Example 12 should be provided to the `calibre -query svdb` command.

Note:

Calibre query commands are executed sequentially. Changing the order of the commands might affect the results.

Example 12 Calibre Query Command File

```
LAYOUT NETLIST DEVICE LOCATION CENTER
# Instructs query server to write net ID to seed polygons
GDS SEED PROPERTY ORIGINAL

# Instructs query server to output further information about the
# device recognition layer to the CCI database.
GDS SEED PROPERTY DEVICE ORIGINAL

#Generates the GDS_MAP layer file.
RESPONSE FILE GDS_MAP
GDS MAP
RESPONSE DIRECT

# The following lines define the property numbers for net names and
# instance names. StarRC expects the NETPROP number to be 5, the
# PLACEPROP number to #be 6, and INSTPROP number to be 7. Do not change.
GDS NETPROP NUMBER 5
GDS PLACEPROP NUMBER 6
GDS DEVPROP NUMBER 7

#Outputs Calibre AGF file for StarRC.
GDS WRITE agf_file

# These commands ensure pin coordinates and proper hierarchy
# in the ideal layout netlist written out by Calibre.
# AGF is the only allowed keyword for LAYOUT NETLIST HIERARCHY.
LAYOUT NETLIST DEVICE LOCATION CENTER
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
LAYOUT NETLIST PIN LOCATIONS YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE netlist_file

# Outputs Calibre ideal layout name map for StarRC.
# The EXPAND_CELLS keyword for LAYOUT NAMETABLE WRITE ensures that the
# lnn file and the netlist have the same hierarchy. The EXPAND_CELLS
# keyword requires Calibre version 2014.3 or later.
```

Feedback

```
LAYOUT NAMETABLE WRITE lnn_file EXPAND_CELLS

# Outputs Calibre net cross-referencing table file; required to run XREF.
# The LNXF keyword requires Calibre version 2014.3 or later.
NET XREF WRITE nxf_file LNXF

# Outputs Calibre instance cross-referencing table for StarRC
# This is required to run XREF.
INSTANCE XREF WRITE ixf_file

# Outputs Calibre cell extents file for StarRC
CELL EXTENTS WRITE extents.txt

# Outputs Calibre top-level ports file for StarRC.
PORT TABLE CELLS DB_CONNECTIVITY WRITE cells_ports_file

# Outputs Calibre device table report file for StarRC.
RESPONSE FILE devtab_file
DEVICE TABLE
RESPONSE DIRECT
```

When creating a query server command file, keep the following points in mind:

- The path of the LVS extraction file must be included in the Calibre query file.

  ```
  LVS SETTINGS REPORT WRITE ./CCI_query/extraction_report_file
  ```

  When you include the LVS extraction report file in the Calibre Connectivity Interface, the StarRC tool does not need to parse the Calibre rulefile, so the Calibre runset is not required. It is the responsibility of the Calibre tool to ensure that the information in the LVS extraction report is accurate and handles directives and conditional statements.

- Additional GDS MAP layer mapping commands to specify specific GDS layer numbers are not required for the query server to output relevant connectivity and device terminal layers to the AGF file. They simply change the AGF output layer number. Using GDS MAP commands in this way is problematic, because they can cause the query server to output layers to a layer number that is already internally mapped to another layer by the Calibre tool. Only the following commands are essential for correct layer mapping in the StarRC tool:

  ```
  RESPONSE FILE GDS_MAP
  GDS MAP
  RESPONSE DIRECT
  ```

- The following commands must be specified to identify the properties that the StarRC tool should obtain from the AGF file:

  ```
  GDS NETPROP NUMBER
  GDS PLACEPROP NUMBER
  GDS DEVPROP NUMBER
  ```

- The following command directs the Calibre tool to report the device center as the device location instead of the default vertex:

  ```
  LAYOUT NETLIST DEVICE LOCATION CENTER
  ```

- If you use Virtuoso Integration with the hierarchical Calibre Connectivity Interface, add the following command to the query server command file:

  ```
  HIERARCHY SEPARATOR |
  ```

  This forces the Calibre tool to use the pipe character (`|`) as the hierarchical separator for compatibility with Virtuoso Integration.

Database Layer Connectivity Inheritance

To simplify LVS runsets for advanced process nodes, the `LVS DB CONNECTIVITY LAYER` statement is used in the LVS runset to pass connectivity information for layers needed for parasitic extraction. This option specifies that a layer derived from a parent layer inherits the layer connectivity from the parent layer and eliminates the need to replicate connectivity for derived layers. To ensure that the ports present in the intersection of parent and child layers are output on child layers, add the `DB_CONNECTIVITY` option to the `PORT TABLE CELLS` query command.

For example, consider the following command:

```
PORT TABLE CELLS WRITE CCI/CCI_DB.port_table
```

Modify the command as follows:

```
PORT TABLE CELLS DB_CONNECTIVITY WRITE CCI/CCI_DB.port_table
```

Using this option only affects layers that are in the `LVS DB CONNECTIVITY LAYER` list. If there are no such layers, the option has no effect.

Feedback

Stripping X Cards in Source Names

The following command directs Calibre to strip X cards in source names:

```
XREF XNAME SOURCE OFF
```

This command strips X card prefixes at each hierarchical level of source net and instance names in the NXF and IXF tables. Such functionality is useful in StarRC gate-level extractions based on hierarchical Calibre LVS runs, in which the StarRC parasitic netlist must back-annotate to an original Verilog netlist that does not contain X card prefixes in each level of hierarchy of a hierarchical net or instance name.

The effect on the StarRC output parasitic netlist of using this Calibre query server command is illustrated by the following SPF example:

A StarRC netlist without the `XREF XNAME SOURCE OFF` command:

```
*|NET XA/XB/XC/net0  0.225231PF
*|I (XA/XB/XC/MM1:D XA/XB/XC/MM1 D B 0 13 175)
R1 XA/XB/XC/MM1:D XA/XB/XC/net0:4 1.2335
Cg1 XA/XB/XC/net0:4 1.63099e-16
```

A StarRC netlist with the `XREF XNAME SOURCE OFF` command:

```
*|NET A/B/C/net0  0.225231PF
*|I (A/B/C/MM1:D A/B/C/MM1 D B 0 13 175)
R1 A/B/C/MM1:D A/B/C/net0:4 1.2335
Cg1 A/B/C/net0:4 1.63099e-16
```

Note:

The `XREF XNAME SOURCE OFF` command should appear in the query server command file before the `NET XREF WRITE` and `INSTANCE XREF WRITE` commands.

The corresponding Calibre query server `XREF XNAME LAYOUT OFF` command should not be used with the StarRC tool. This is because the layout netlist generated by Calibre always contains X cards regardless of the setting of `XREF XNAME LAYOUT`, and the StarRC tool requires consistency between the NXF and IXF layout names and the Calibre-generated layout netlist.

This command is not applicable to and produces no differences for StarRC runs based on Calibre flat LVS runs.

See the Calibre documentation for further details.

Feedback

Multifinger Device Support in the Calibre Connectivity Interface

The Calibre Connectivity Interface provides multifinger access resistance calculation by creating pins for each finger. To use this feature, add the following Calibre query command:

```
GDS SEED PROPERTY DEVICE ORIGINAL
```

With this command in the query command file, the StarRC tool initiates automatic pin detection to obtain the pin location for each terminal of the device. When there is more than one polygon for one terminal, the automatic pin detection code generates a pin location for each terminal polygon, providing more accurate resistance extraction results.

It is important to preserve the cell hierarchy of multifinger devices during LVS. The seed layer (device recognition layer) and the finger terminal shapes should be at the same level of the hierarchy to facilitate multifinger detection in the StarRC tool. To preserve cell hierarchy during LVS, use the PEX PRESERVE CELLS LIST command in the Calibre tool to define a cell that contains multifinger devices.

Note:

The PEX PRESERVE CELLS LIST command is available in the Calibre LVS tool beginning with version 2019.1.

Optional Layout Netlist Query Commands

The following topics describe specific flows for the Calibre Connectivity Interface:

- The Push Down Separate Properties (PDSP) Flow
- The Push Down Back-Annotation (PDBA) Flow
- Using the Calibre Query Flow With the NanoTime Tool
- Error Conditions for the PDSP and PDBA Flows

The Push Down Separate Properties (PDSP) Flow

The PDSP flow is the recommended back-annotation flow. In the PDSP flow, the separate properties are written to a file during query output generation, not during the LVS run. The StarRC tool automatically reads the file from the query output.

To use the PDSP flow, insert these commands into the query command file:

```
LAYOUT NETLIST DEVICE LOCATION CENTER
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
```

```
LAYOUT NETLIST SEPARATED PROPERTIES NO
LAYOUT NETLIST PIN LOCATIONS NO
LAYOUT NETLIST DEVICE TEMPLATES YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE nl
LAYOUT SEPARATED PROPERTIES WRITE CCI/pdsp
```

The Push Down Back-Annotation (PDBA) Flow

The PDBA flow is an older back-annotation flow. In the PDBA flow, the separate properties are written to a file during the LVS run. The file must be referenced in the StarRC command file by using the `CALIBRE_PDBA_FILE` command.

To use the PDBA flow, insert these commands into the query command file:

```
LAYOUT NETLIST DEVICE LOCATION CENTER
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
LAYOUT NETLIST SEPARATED PROPERTIES NO
LAYOUT NETLIST PIN LOCATIONS NO
LAYOUT NETLIST DEVICE TEMPLATES YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE nl
```

Using the Calibre Query Flow With the NanoTime Tool

If you are planning to use the NanoTime transistor-level static timing analysis tool, insert the following command at the end of the query command file immediately before you execute the Calibre query:

```
REDUCTION DATA WRITE source.nrdw SOURCE NET FLAT WITH XREF
```

The file named source.ndrw contains SEN and DEEPSHORT lines that are read by the NanoTime tool to ensure accurate back-annotation of parasitics for swapped devices on equivalent nets. In the NanoTime tool, use the `set_lvs_equivalent_nets -net_combined_report` command to specify the file name.

For more information, see the *NanoTime User Guide*.

Note:

This feature requires NanoTime version M-2017.06-SP2 or later and Calibre version 2017.4 or later.

Feedback

Error Conditions for the PDSP and PDBA Flows

The StarRC tool issues warning or error messages if the tool finds unsupported commands in the Calibre query command file. For error messages, extraction also stops. The affected Calibre query commands are shown in Table 45.

Table 45 Unsupported Calibre Query Commands

Command	Setting	StarRC result
LAYOUT NETLIST PIN LOCATIONS LAYOUT NETLIST DEVICE TEMPLATES (used together in PDSP or PDBA flows)	NO (default) YES	allowed
LAYOUT NETLIST PIN LOCATIONS LAYOUT NETLIST DEVICE TEMPLATES (used together in PDSP or PDBA flows)	NO NO (default)	warning
LAYOUT NETLIST PIN LOCATIONS LAYOUT NETLIST DEVICE TEMPLATES (used together in PDSP or PDBA flows)	YES YES	warning
LAYOUT NETLIST PIN LOCATIONS LAYOUT NETLIST DEVICE TEMPLATES (used together in PDSP or PDBA flows)	YES NO	warning
LAYOUT NETLIST SEPARATED PROPERTIES (used in PDSP or PDBA flows)	NO (default) YES	allowed error
LAYOUT NETLIST HIERARCHY (all flows including PDSP and PDBA flows)	AGF ALL (default) FLAT HCELL	allowed error error error

Feedback

Preparing the StarRC Command File

Instructions for preparing a StarRC command file are found in this section. Options to be included in this file are explained.

Commands for Hercules Flows

To run the transistor-level Hercules flow, you must use the following commands:

```
MILKYWAY_DATABASE:       LIB_NAME
MILKYWAY_EXTRACT_VIEW:   YES
BLOCK:                   BLOCK
SKIP_CELLS:              !*
TCAD_GRD_FILE:           technology.nxtgrd
MAPPING_FILE:            mapping_file_name
```

`MILKYWAY_DATABASE`: *`LIB_NAME`* is the path to the directory having the XTR view generated by Hercules. It should be the same as the *`LIB_NAME`* in the `WRITE_EXTRACT_VIEW` command in the Hercules runset. However, you are allowed to set a new path if you have moved the directory to another place. Note that the default for `SKIP_CELLS` is "*" and you must make sure that you set it as "!*" to extract down to the transistor level.

Commands for IC Validator Flows

To run the transistor-level IC Validator flow, you must use the following commands:

```
ICV_RUNSET_REPORT_FILE: file_name
BLOCK:                  BLOCK
SKIP_CELLS:             !*
TCAD_GRD_FILE:          technology.nxtgrd
MAPPING_FILE:           mapping_file_name
```

The StarRC tool obtains information such as connection information, the location of LVS COMPARE output, device pin and properties information, and the location of the extract view from the runset report file specified by the `ICV_RUNSET_REPORT_FILE` command. The default for `SKIP_CELLS` is "*" and you must make sure that you set it as "!*" to extract down to the transistor level.

Commands for Calibre Connectivity Interface Flows

To run the transistor-level Calibre Connectivity Interface flow, you must specify the following options:

```
BLOCK: block_name
SKIP_CELLS: !*
```

Feedback

```
TCAD_GRD_FILE: technology.nxtgrd
MAPPING_FILE: mapping_file_name
CALIBRE_QUERY_FILE: calibre_query_command_file
```

Calibre options are explained in The Calibre Connectivity Interface Flow.

Other StarRC Commands

The following commands are also important for StarRC command file preparation.

NETLIST_FORMAT

The NETLIST_FORMAT command defines the structure and format of the output parasitic netlist:

```
NETLIST_FORMAT: SPF | SPEF | NETNAME | OA  | STAR
```

For transistor-level extraction, the StarRC tool supports the SPF, SPEF, OA, STAR, and NETNAME formats. You can use the NETLIST_CONNECT_SECTION: NO and NETLIST_NODE_SECTION: NO commands to suppress *|I and *|S sections.

IGNORE_CAPACITANCE

This command disallows certain types of device-level capacitances from being extracted:

```
IGNORE_CAPACITANCE: ALL | DIFF | NONE
```

You can ignore both gate and diffusion capacitance with the IGNORE_CAPACITANCE command.

If you set IGNORE_CAPACITANCE: ALL, both overlap and sidewall capacitances between gate and SUBSTRATE is ignored. All other coupling effects to the gate poly node from other conducting layers are considered. If you set IGNORE_CAPACITANCE to ALL or DIFF, you can ignore the junction capacitance between diffusion and SUBSTRATE.

The IGNORE_CAPACITANCE command is runset-layer-based. This means that each layer in the runset (even if multiple runset layers are mapped to a single ITF layer) is processed individually. If the runset defines a PMOS device that uses NWELL as the BULK layer, NWELL must be the only layer under the device for the IGNORE_CAPACITANCE command to work.

If the runset contains a connected copy of NWELL that is also present under every PMOS device, the IGNORE_CAPACITANCE command has no effect (because the NWELL copy is not tagged as a MOS BULK layer). The runset should not contain connected copies of BULK.

Feedback

XREF

This command determines which set of names to report for StarRC netlist generation and analysis flows and which devices and nets to retain if both layout names and cross-referenced schematic names are present.

```
XREF: NO | YES | COMPLETE
```

The XREF command enables cross-referencing of the parasitic netlist for LVS-based extraction flows. Nets, devices, and cell instances are written in the parasitic netlist using schematic names, according to the functionality of modes YES and COMPLETE.

Note:

All three XREF options are available in the Hercules flow. However, only the YES and NO options are available in the Calibre Connectivity Interface flow.

If you set XREF:COMPLETE, all the nets that are not cross-referenced are ignored. Only successfully matched nets and instances are included in the output file.

If you want to use the SKIP_CELLS command in cross-referencing flows, be sure that the skip cells are EQUIV points (in the Hercules or IC Validator flows) or hcells (in the Calibre flow) that were successfully matched during LVS. Those cells that are not matched during LVS cannot be properly skipped.

Because the StarRC tool gets cross-reference information from the evaccess and compare directories, do not remove those directories (in the Hercules or IC Validator flow) before the StarRC run is over.

COMPARE_DIRECTORY and EVACCESS_DIRECTORY

The COMPARE_DIRECTORY command specifies the location of the Hercules LVS COMPARE output. The EVACCESS_DIRECTORY command specifies the location of the Hercules LVS EvAccess database.

```
COMPARE_DIRECTORY: PATH_TO_compare_DIRECTORY
EVACCESS_DIRECTORY: PATH_TO_evaccess_DIRECTORY
```

In XREF:YES or COMPLETE runs, the StarRC tool gets the cross-reference information and schematic netlist information from the compare and evaccess directories. If the paths to the original compare and evaccess directories have changed, you have to set the COMPARE_DIRECTORY and EVACCESS_DIRECTORY commands with the new paths. These options can be skipped if they are the same as the Hercules run.

Interconnect Technology Format File

This section explains how to prepare the Interconnect Technology Format (ITF) file. For more information, see Flows for Process Characterization.

Feedback

Preparing the ITF File

Preparing an ITF file for transistor-level extraction is similar to cell-level extraction; the difference is that transistor-level ITF files have the connectivity information down to the diffusion or SUBSTRATE layers. Both planar as well as nonplanar process files can be created.

The transistor-level ITF file can be either planar, using a single poly layer for both field and gate poly, or nonplanar, using separate field and gate poly layers. Using a planar process for transistor-level extraction allows for faster nxtgrd file generation time relative to the nonplanar process, because one less `CONDUCTOR` layer exists in a planar ITF file. However, the choice is optional and should be dictated by the parameters of the actual physical process.

An ITF file contains via layer information and you need to be careful when defining them. Because a via layer can connect only two conducting layers, you must separate a poly contact from a SUBSTRATE contact (and diffusion contact, if any).

Limitations

Covertical layers (such as gate and field poly) should be vertically overlapping. Thus if the bottom of the field poly layer is higher than the top of the gate poly layer (as happens when the field oxide is too thick or the gate poly is too thin), they are not connected to each other by mere touch. In this case, you need to modify your runset or rule file, mapping file, and ITF file to make a dummy via layer connecting the gate and field poly.

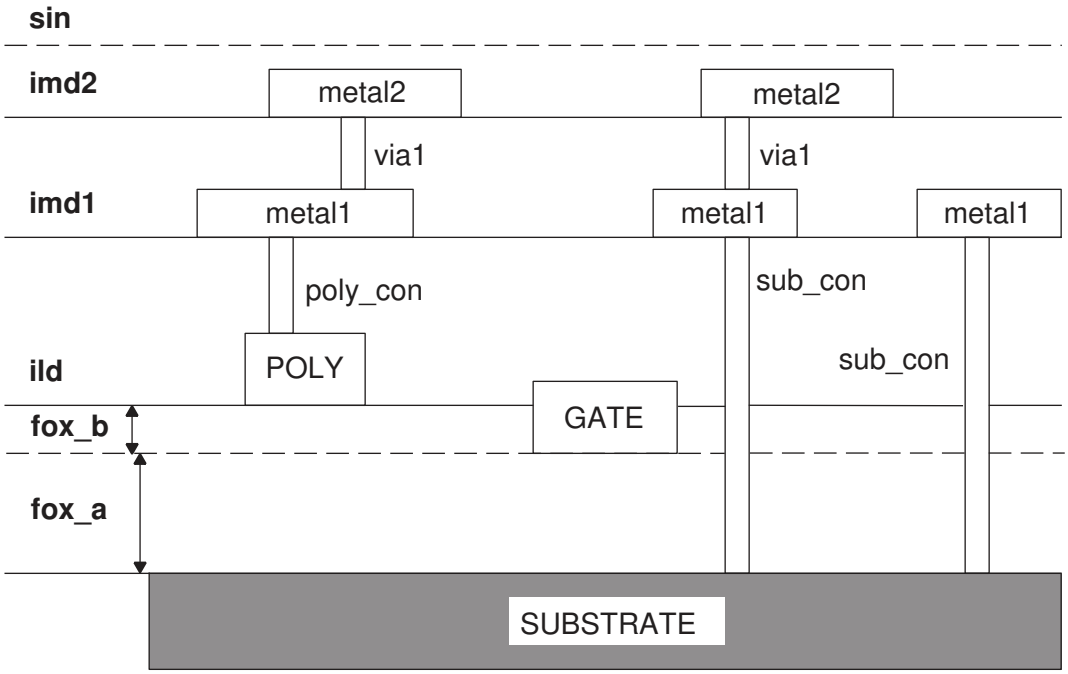
ITF File Example

The following example shows an ITF file that corresponds to the example in the Interconnect Technology Format File section. The process cross section is shown in Figure 98.

```
TECHNOLOGY=add4_2m1p

DIELECTRIC sin {THICKNESS=0.70 ER=7.9}
DIELECTRIC imd2 {THICKNESS=1.00 ER=4.2}
CONDUCTOR metal2 {THICKNESS=0.50 WMIN=0.35 SMIN=0.35 RPSQ=0.07}
DIELECTRIC imd1 {THICKNESS=1.05 ER=4.2}
CONDUCTOR metal1 {THICKNESS=0.35 WMIN=0.30 SMIN=0.30 RPSQ=0.08}
DIELECTRIC ild    {THICKNESS=0.70 ER=4.1}
CONDUCTOR poly   {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_b {THICKNESS=0.10 ER=3.9}
CONDUCTOR gate {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_a {THICKNESS=0.25 ER=3.9}

VIA via1     {FROM=metal1    TO=metal2 AREA=0.09 RPV=1}
```

```
VIA poly_con {FROM=poly      TO=metal1 AREA=0.04 RPV=12}
VIA sub_con {FROM=SUBSTRATE TO=metal1 AREA=0.04 RPV=16}
```

Figure 98 Process Cross Section



Transistor-Level Extraction Limitations

The following general limitations apply:

- Covertical layers should be overlapping vertically.
- Only `XREF: YES` is supported in the Calibre Connectivity Interface flow.
- In the Calibre Connectivity Interface flow, each device extraction command listed in the rule file must have a unique model name, to allow the StarRC tool to properly distinguish devices.

The following limitations apply to cross-referencing flows:

- `DETECT_PERMUTABLE_PORTS` is not supported.
- `PUSH_DOWN_DEVICES` (Hercules) is not handled correctly.
- `PUSH_DOWN_PINS: FALSE` (Hercules) should be set when possible to achieve best results with `XREF`.

Feedback

- In XREF:COMPLETE flows in which m(schematic) : n(layout) or m:1 device merging occurs, the StarRC tool considers nets that connect to device terminals as ideal nets. This is because no method exists by which to distribute parasitics from a single layout net across the multiple schematic nets that are included in the XREF:COMPLETE netlist.
- In XREF:COMPLETE netlists for which multistage m:n merging occurs for designed passive devices, NETLIST_PASSIVE_PARAMS parameters might not appear in the parasitic netlist. No method exists by which to annotate layout device properties from N layout devices onto M schematic devices, because no direct one-to-one correlation can be established among the layout and schematic devices.

10

Special-Purpose Features

pending what to add

This chapter describes additional features available in the StarRC tool.

For more information, see the following topics:

- Parasitic Netlist Checker
- GPD Netlist Checker
- Clock Net Inductance Extraction
- Mutual Inductance Extraction
- Standalone Reducer
- Feedthrough Nets
- Long Ports
- Via Coverage

Feedback

Parasitic Netlist Checker

The StarRC tool provides a parasitic netlist checker that operates on an SPF file to verify the output of a netlist in a transistor-level flow.

To verify the output of the parasitic netlist, use the `check_parasitics_consistency` command.

Syntax

```
check_parasitics_consistency
   file_name
   [-output output_file_directory]
   [-skip_nets list_of_nets]
   [-cap_rel relative_cap_error_threshold]
   [-resistor_threshold threshold_value]
   [-node_cap_threshold_abs absolute_cap_threshold]
   [-node_cap_threshold_rel relative_cap_threshold]
   [-node_cap_threshold_operation AND | OR]
   [-ignore_ln_ports]
   [-ignore_ln_nets]
   [-rename_file]
```

Table 46 lists the following information:

- Options and arguments with descriptions
- Checks performed by each option and the name of the output file generated or error message issued by the tool

Example 13 Using the check_parasitics_consistency Command

```
starrc_shell> check_parasitics_consistency xyz.spf -skip_nets "VSS VDD"
 -node_cap_threshold_operation and \
    -rename_file -ignore_ln_ports -ignore_ln_nets -cap_rel 0.01
 -resistor_threshold 10 -node_cap_threshold_abs 0.5 \
    -node_cap_threshold_rel 0.3
```

Table 46 Checks Performed by the Options of check_parasitics_consistency

Option and Argument	Description	Checks performed	Output file and error message
`file_name`	Name of a parasitic netlist file		
`-output`	Specifies the name of the output file directory Default: current directory		

Feedback

Table 46 Checks Performed by the Options of check_parasitics_consistency (Continued)

Option and Argument	Description	Checks performed	Output file and error message
`-cap_rel`	Specifies relative error threshold for capacitance Default: 0.02	Total capacitance adds up to individual capacitance elements	*check_tcap.rpt*: Reports nets with inconsistent threshold capacitance. For example, `check_parasitics_consistency -cap_rel 0.01`
`-skip_nets`	Lists nets that should be skipped during connection check	No subnodes are missing for nodes used in RC network if you use the `NETLIST_NODE_SECTION: YES` command	*missing_subnode.rpt*: Reports missing subnodes as some nets, such as power nets, might not have connection section. For example, `check_parasitics_consistency -skip_nets "VSS VDD"`
`-node_cap_threshold_abs`	Reports the lumped capacitance tied to a node if the capacitance value is greater than the absolute threshold Default: 1 PF	Large lumped capacitance tied to a node	*large_lump_cap.rpt*: Reports capacitance of a node if the value of node capactance is larger than the absolute threshold and relative capacitance of node capacitance to the total capacitance is larger than the relative threshold.
`-node_cap_threshold_rel`	Reports the lumped capacitance tied to a node if the capacitance value is greater than the relative threshold Default: 0.3		
`-node_cap_threshold_operation`	Specifies the use of AND or OR for a node to check capacitance threshold Default: AND		
`-ignore_ln_nets`	Ignores trivial nets listed in the *large_lump_cap.rpt* and during total capacitance check		
`-resistor_threshold`	Reports resistors above the specified threshold	Identify resistors above the threshold when `-resistor_threshold` is set	*large_resistor.rpt*: Reports resistors above the specified threshold.

Feedback

Table 46 Checks Performed by the Options of check_parasitics_consistency (Continued)

Option and Argument	Description	Checks performed	Output file and error message
`-ignore_ln_ports`	Ignores trivial ports when checking instance pins	Non-port net and instance section should be consistent in standard parasitic format (SPF) netlists	*missing_instance.rpt*: Reports missing instance sections after checking if the instance pin name of non-port nets exists in the instance section and each defined name has a non-port net in the instance section.
`-rename_file`	Renames the parasitic file to <filename>.bad if any check fails		
		Opens	*check_open.rpt*: Ensures there are no resistor segments floating in a design. When subnodes of a resistor network connect to each other only, this forms a resistor segment.
	.	Duplicate subnode	*duplicate_subnode.rpt*: Ensures there are no duplicate subnodes when there are multiple subnodes with the same location on the same layer. The tool performs this check only if you have used the `EXTRA_GEOMETRY_INFO` command.
		Each statement in the netlist should start with one of the following characters: • R, C, *|[I], *|[P], *|[S],*|[O], or *|[P] No other characters are allowed, but * is allowed to use for comments	*unexpected_lines.rpt*: Verifies each line or statement in the netlist and reports lines or statements that are not beginning with the specified characters.

Feedback

Table 46 Checks Performed by the Options of check_parasitics_consistency (Continued)

Option and Argument	Description	Checks performed	Output file and error message
		StarRC run is successful but the SPF file is incomplete by missing the *|NET or *D_NET section	The tool issues the following error message: `ERROR: spf file is not complete. *|NET section is missing`
		Validity of a SPF netlist (should include the end section)	The tool issues the following error message: `ERROR: spf file is not complete. ".ENDS" is missing`

See Also

- SPF_CHECKS

Feedback

GPD Netlist Checker

The StarRC tool provides a GPD netlist checker that operates on a GPD to verify the output of a netlist in a gate-level flow. You can use the GPD netlist checker in standalone StarRC, distributed processing, and StarRC GPD flows to do the following tasks with gate-level designs only:

- Perform consistency checks directly on the GPD without generating netlist files
- Validate parasitic information earlier in the StarRC flow

To use the GPD netlist checker in the flows, you need to create a configuration file with the user-defined options. If you do not specify any options, the tool applies the default as required.

Note:

The tool issues the SX-3991 error message if options are specified with invalid values.

You must specify the following options in the configuration file:

```
config file
   -tcap_rel rel_total_cap_error
   -node_cap_threshold_abs abs_node_cap_threshold
   -node_cap_threshold_rel rel_node_cap_threshold
   -node_cap_threshold_operations node_cap_checker
   -tcap_trend_order total_cap_corner_order
   -tcap_trend_delta total_cap_threshold
   -tcap_trend_delta_reversed total_cap_threshold_reversed
   -entries number_report_mesg
   -check_coupling_symmetry yes or no
```

Table 47 lists the following information required to use the GPD netlist checker:

- Options and arguments with descriptions
- Checks performed by each option and the name of the output file generated or error message issued by the tool

You can use the GPD netlist checker in the following flows:

- In the standalone StarRC flow, specify the `StarXtract` command at the operating system prompt as follows:

```
% StarXtract -check_gpd_consistency gpd_dir config_file \
   -keep_checker_star
```

Feedback

- In the standalone StarRC distributed processing flow, specify the `StarXtract` command at the operating system prompt as follows:

```
% StarXtract -check_gpd_consistency gpd_dir config_file \
   -keep_checker_star -dp_run -localhost -set_num_cores
```

- In the StarRC flow, use the `GPD_CHECKS` command.

Table 47 Checks performed by the options specified in configuration file

Option and Argument	Description	Checks performed	Output file and error message
`-check_gpd_consistency`	Verifies the output of the netlist file		
`-keep_checker_Star`	Specifies the star directory, retains the file if specified		
`gpd_dir`	Specifies the path to the GPD directory		
`-dp_run`	Uses the GPD netlist checker in the distributed processing mode		
`-localhost`	(Optional) Specifies the number of local host		If `-localhost` and `-set_num_cores` are not specified with the `StarXtract` command, the GPD netlist checker uses the `STARRC_DP_STRING` command specified in technology files.
`-set_num_cores`	(Optional) Specifies the number of cores		
`config file`	(Optional) Specifies the path to a configuration file		
`-tcap_rel`	Specifies relative error threshold for total capacitance Default: 0.0001	Capacitance of individual elements adds to the total capacitance	*check_tcap.rpt*: Reports nets with inconsistent threshold capacitance. For example, `check_parasitics_consistency -cap_rel 0.01`

Feedback

Table 47 Checks performed by the options specified in configuration file (Continued)

Option and Argument	Description	Checks performed	Output file and error message
`-node_cap_ threshold_abs`	Reports the lumped capacitance tied to a node if the capacitance value is greater than the absolute threshold Default: 10,000 PF	Large lumped capacitance tied to a node	*large_lump_cap.rpt*:: Reports the following information: • Capacitance of a node if the value of node capactance is larger than the absolute threshold and • Relative capacitance of the node capacitance to the total capacitance is larger than the relative threshold
`-node_cap_ threshold_rel`	Reports the lumped capacitance tied to a node if the capacitance value is greater than the relative threshold Default: 0.3		
`-node_cap_ threshold_operation`	Specifies the use of AND or OR gate for a node to check capacitance threshold Default: AND		
`-tcap_trend_order`	Specifies corner order trend for total capacitance trend analysis		
`-tcap_trend_delta`	Specifies trend analysis threshold for each corner for total capacitance Default: 0.0		
`-tcap_trend_delta_reversed`	Specifies trend analysis threshold for each corner, in the opposide-side direction of the specified comparator for total capacitance Default: 0.0		
`-entries`	Specifies a number to report messages in a file Default: 1000		

Feedback

Table 47 Checks performed by the options specified in configuration file (Continued)

Option and Argument	Description	Checks performed	Output file and error message
`-check_coupling_symmetry`	Specifies whether to check coupling symmetry in the GPD directory Default: no		`asymmetric_cap.rpt`: Reports the asymmetric capacitance in the GPD directory. If this option is set to `yes`, the `-dp_run` option in the command line issues an error message

See Also

- The Parasitic Database or GPD
- The StarXtract Command
- GPD_CHECKS
- SPF_CHECKS

Feedback

Clock Net Inductance Extraction

Inductance is important at high operating frequencies for long nets, such as clock nets. Inductance effects include signal overshoot and undershoot, bumps in the waveform, and increased clock skew.

You can use the StarRC tool to perform clock net inductance extraction. Clock net inductance extraction requires targeted runs, separate from standard full-chip or signoff extraction runs. The output is a DSPF netlist containing full RLC information for the selected nets. This netlist is typically used as input for HSPICE or other circuit simulation tools. Two levels of analysis are available: standard and advanced.

You can use any foundry-provided nxtgrd file for clock net inductance extraction. All process variation effects modeled within the nxtgrd file are acceptable.

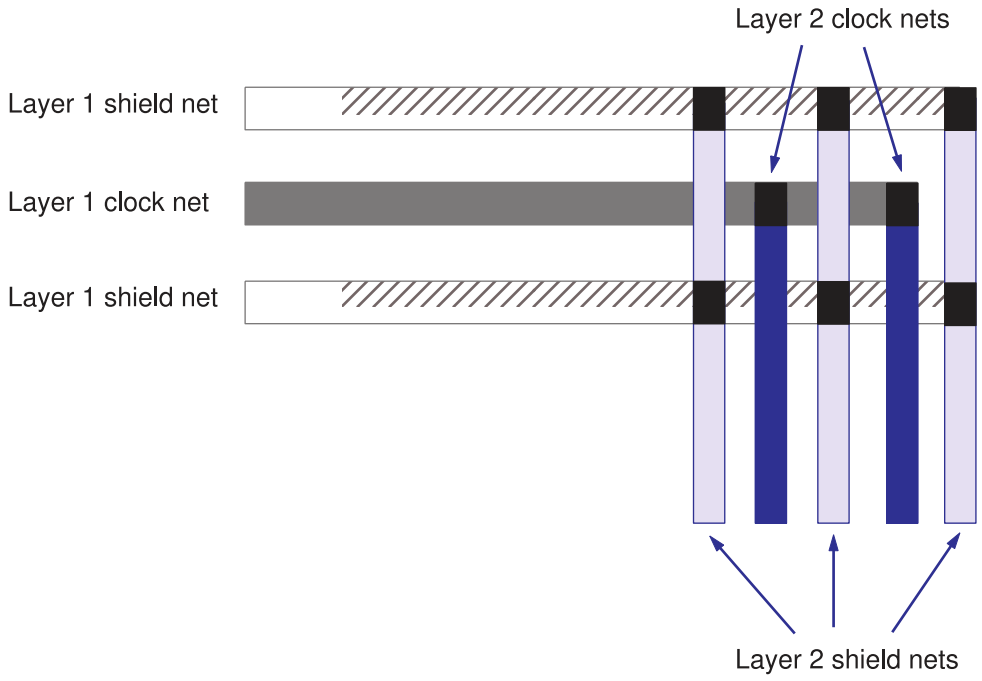
The following usage notes apply:

- Only DSPF netlists are supported for output.
- The design must be a gate-level Fusion Compiler, IC Compiler II, Milkyway, or LEF/DEF design.

An example of a shielded clock net is shown in Figure 99. Shield nets appear on both sides of the clock net. A design might contain shielded clock nets on more than one layer.

Feedback

Figure 99 Clock Nets With Shielding



Standard Clock Net Inductance Extraction

Clock net inductance analysis is disabled by default. To enable it, set the `CLOCK_NET_INDUCTANCE` command to `YES`.

To use standard clock net inductance analysis, use the `CLOCK_NET_FREQUENCY` and `CLOCK_NET_INDUCTANCE_LAYERS` commands. Shielding is modeled as follows:

- A clock net must be shielded by power or ground nets on the same layer. Signal nets cannot act as shielding nets.
- The current return is assumed to occur exclusively through the power or ground shield nets.
- Inductive coupling between a clock net and its shield nets is ignored.
- Inductive coupling between clock domains is ignored because the clock nets are assumed to be fully shielded.
- Fill polygons between a clock net and its shield nets are allowed.

Feedback

- Shield nets are modeled to be no farther than five microns away from the clock net. In other words, the modeled shield distance is either the actual spacing or five microns, whichever is smaller. In some designs, the shield distance might vary along the length of the clock net.

An example of a command file for standard clock net inductance extraction is as follows:

```
EXTRACTION: RC
NETS: list_of_cnets
CLOCK_NET_INDUCTANCE: YES
(optional) CLOCK_NET_FREQUENCY: cnet_freq
(optional) CLOCK_NET_INDUCTANCE_LAYERS: cnet_layers
REDUCTION: YES | NO
NETLIST_FORMAT: SPF
SIMULTANEOUS_MULTI_CORNER: NO
POWER_EXTRACT: NO
```

Analysis requirements are as follows:

- Only the `RC` setting of the `EXTRACTION` command is allowed. If the command is not explicitly defined, the default is `RC`.
- Only the `YES` and `NO` settings of the `REDUCTION` command are allowed. If the command is not explicitly defined, the default is `YES`.
- If the `CLOCK_NET_INDUCTANCE_LAYERS` command is used, nets listed in the `NETS` command are extracted only if they are found on the specified layers.
- Simultaneous multicorner extraction and power net extraction must be disabled.

Advanced Clock Net Inductance Extraction

You can enable optional features of clock net inductance extraction by setting the `CLOCK_NET_ADVANCED_MODEL` command to `YES`, in addition to setting the `CLOCK_NET_INDUCTANCE` command to `YES`. The analysis requirements for standard clock net inductance extraction also apply to advanced clock net inductance extraction.

Frequency Effects

Inductance decreases with frequency, while resistance increases with frequency. You can model these effects by setting the following commands:

- If the `CLOCK_NET_ADVANCED_MODEL` command is set to `NO` (the default), the StarRC tool calculates one resistance value at zero frequency and one inductance value for each net segment at the frequency specified by the `CLOCK_NET_FREQUENCY` command.
- If the `CLOCK_NET_ADVANCED_MODEL` command is set to `YES` and the `HIGH_CLOCK_NET_FREQUENCY` command is not set, the tool calculates one resistance

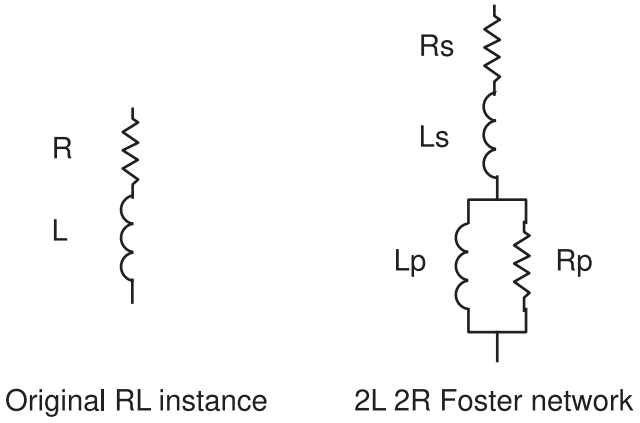
Feedback

value and one inductance value for each net segment at the frequency specified by the CLOCK_NET_FREQUENCY command.

- If the CLOCK_NET_ADVANCED_MODEL command is set to YES and the HIGH_CLOCK_NET_FREQUENCY command is set to a value larger than the CLOCK_NET_FREQUENCY command value, the tool calculates frequency-dependent resistance and inductance values for each net segment. The result is a netlist that can be used for frequency analysis in downstream simulation tools. The model is valid between the two specified frequencies.

For the frequency-dependent model, each resistor-inductor (RL) instance in the original design is replaced by a 2L 2R Foster network, as shown in Figure 100.

Figure 100 Foster Inductance Model For Frequency Dependence



Shielding Options

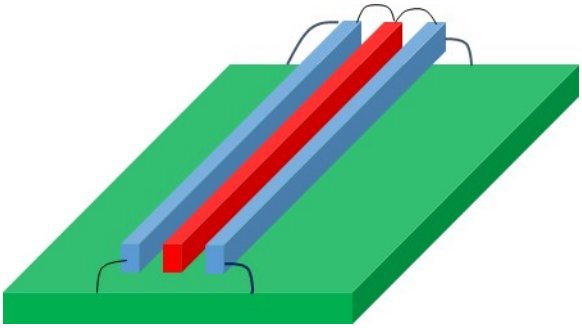
The advanced shielding options are as follows:

- You can model return current that occurs in shield conductors beyond the immediately adjacent shield conductors by using the CLOCK_NET_SHIELD_EXT_FACTOR command to specify the relative conductivity of the external shields. The default is 0.5. Figure 101 illustrates the use of external shields.
- You can optionally specify a layer to use as a ground layer as an additional path for return current by using the CLOCK_NET_INDUCTANCE_GND_LAYER command. Figure 102 illustrates a ground plane.

Feedback

Figure 101 External Shields For Clock Net Inductance Extraction

Figure 102 Ground Plane For Clock Net Inductance Extraction



Feedback

Mutual Inductance Extraction

The StarRC tool can analyze mutual inductance on small blocks or on portions of larger circuits. Inductance extraction is performed as an additional step in a transistor-level extraction. The tool performs extraction as directed by the command file, then analyzes inductance on a selected portion of the design, as specified by the `INDUCTANCE_*` commands. The inductance results are then inserted into the output SPF file.

Inductance analysis is based on the FastHenry inductance field solver tool. In this mode, the tool generates an unreduced netlist that contains the physical sizes of resistors and the node locations.

By default, the StarRC tool does not analyze inductance. To enable mutual inductance analysis, include the `INDUCTANCE_MODE` command in the StarRC command file.

For inductance analysis, the StarRC tool automatically sets the following commands:

- `REDUCTION: NO`
- `NETLIST_NODE_SECTION: YES`
- `NETLIST_MERGE_SHORTED_PORTS: NO`
- `NETLIST_TAIL_COMMENTS: YES`
- `EXTRA_GEOMETRY_INFO: NODE`
- `PRINT_SILICON_INFO: YES`

The following usage notes apply:

- The runtime and memory requirements increase with the number of inductors. For best results, limit the number of inductors to 10,000 or fewer.
- Power nets often serve as current return paths. For best inductance accuracy, set the `POWER_EXTRACT` command to `YES`. If you analyze inductance without power net extraction, the resulting inductance values tend to be overestimated.
- In cases where multiple vias exist between two conductor shapes, the analysis might create many small inductors between the vias. Set the `MERGE_VIAS_IN_ARRAY` command to `YES` to reduce the number of segments and improve analysis performance.
- Simultaneous multicorner analysis is not supported.
- This feature is mutually exclusive with clock net inductance analysis.
- Only the DSPF output netlist format is supported. Set the `NETLIST_FORMAT` command to `SPF` in the extraction command file.

The StarRC tool offers two types of mutual inductance analysis

- Single-frequency analysis

  Specify a frequency with the INDUCTANCE_FREQUENCY command, the nets of interest with the INDUCTANCE_SELECT_NET command, and the bounding box and layer information with the INDUCTANCE_SELECT_BB command.

- S parameter analysis

  Specify a frequency range with the INDUCTANCE_S_PARAM_FREQUENCY command and the bounding box and layer information with the INDUCTANCE_SELECT_S_BB command.

The following commands apply to both single-frequency and S parameter analysis:

- If you select output in terms of reluctance by setting the INDUCTANCE_MODE command to RELUCTANCE, the tool first performs analysis in terms of inductance, then converts the results to reluctance. The INDUCTANCE_REL_THRESHOLD command improves performance by dropping relatively unimportant terms from the conversion. If you increase the value, runtime is reduced at the cost of accuracy.
- The INDUCTANCE_MIN_LENGTH command specifies the minimum length for a resistor to be included in mutual inductance analysis. Reducing the number of shapes to be included in inductance analysis improves runtime and memory performance.
- The INDUCTANCE_NINC command specifies the number of filaments in which to divide metal segments for skin depth and proximity effect calculations. As frequency increases, the skin depth decreases. Therefore, analysis at higher frequencies requires a finer analysis grid.

  The StarRC tool automatically sets this parameter. Increasing this parameter manually increases accuracy at the cost of runtime and capacity. For most applications, use the default set by the tool.

- The INDUCTANCE_SELECT_LAYER command specifies the layers to be considered in mutual inductance analysis.

  If you specify layers with both the INDUCTANCE_SELECT_LAYER command and the INDUCTANCE_SELECT_BB command, only the layers that appear in both commands are selected for inductance analysis (equivalent to a logical AND operation).

  The same rule applies if you use both the INDUCTANCE_SELECT_LAYER command and the INDUCTANCE_SELECT_S_BB command. However, the INDUCTANCE_SELECT_BB command and the INDUCTANCE_SELECT_S_BB command have no effect on each other because they apply to different types of analysis.

Feedback

Single-Frequency Inductance Analysis

The following is an example of the commands used for single-frequency inductance analysis:

```
INDUCTANCE_MODE: INDUCTANCE
INDUCTANCE_SELECT_BB: -36.5 12 -6.4 43.3
INDUCTANCE_SELECT_LAYER: M13 M12 M11
INDUCTANCE_REL_THRESHOLD: 1
INDUCTANCE_MIN_LENGTH: 0.7
INDUCTANCE_SELECT_NET: vo_m
INDUCTANCE_FREQUENCY: 1
POWER_EXTRACT: YES
MERGE_VIAS_IN_ARRAY: YES
```

The following is an example of a netlist resulting from single-frequency inductance analysis:

```
*|NET vo_m 0.0309027PF
*|P (vo_m B 0 0 0 0 0)
*|I (xrrosm1 MINUS xroosm1 MINUS B 0 0 0 0 0 0)
*|I (Xindm.xl17.xralia.PLUS Xindm.xl17.xralias PLUS B 0 0 0 0 0 0)
*|I (Xindm.xl34.xralia.PLUS Xindm.xl34.xralias PLUS B 0 0 0 0 0 0)
R0 vo_m:47 vo_m 0.45213
R1 vo_m416 xrrosm1:MINUS 21.9044
C0 xrrosm1:MINUS 0.297136e-16
R2 vo_m:17 r_l_23 0.0417795 $l=1.6753 $si_w=0.9804 $lvl=18
L23 r_l_23 Xindm.xl17.xralias:PLUS 4.47121e-13
R3 vo_m:3 r_l_4 0.311744 $l=12.1633 $si_w=0.9804 $lvl=18
L4 r_l_4 Xindm.xl34.xralias:PLUS 6.86971e-12
...
K_0 L0 L1 -0.0030765
K_1 L0 L2 0.438044
K_2 L0 L3 0.00631556
...
```

S Parameter Analysis

For S parameter analysis, the tool considers a three-dimensional box defined by the x- and y-coordinates of the bounding box (specified in the `INDUCTANCE_SELECT_S_BB` command) and the layers of interest (specified in either the `INDUCTANCE_SELECT_S_BB` or `INDUCTANCE_SELECT_LAYER` command).

The tool automatically creates S-parameter ports by selecting the node inside the 3-D box closest to the location where a branch enters the box. Instance ports inside the 3-D box also become ports for the model. The model includes inductive couplings within the 3-D box, but no coupling between separate 3-D boxes or to conductors outside of the 3-D box. To be considered, a metal segment must be on a specified layer and must be longer than the value specified in the `INDUCTANCE_MIN_LENGTH` command.

The model includes capacitive couplings within the 3-D box. Capacitive coupling to conductors outside the box are grounded on both ends.

For S parameter analysis, the output netlist presents the information as follows:

```
S_index n1 n2 n3 ... Gnd  NAME=S_M_index
.model S_M_index S N=num_ports TSTONEFILE=outfile
```

In this example, n1, n2, and so on are the port nodes and num_ports is the total number of ports. The ports are automatically generated by the StarRC tool. Gnd is the ground or reference node. Index is an automatically generated index number.

The output file format uses Touchstone version 2.0. For more information, see general information about this open-source format.

Feedback

Standalone Reducer

The StarRC tool provides a standalone reducer that operates on SPEF, SPF, DSPF files, or GPD directory. The reducer removes nodes one at a time until the delay change exceeds a user-specified value. The reducer preserves point-to-point resistance, net-to-net coupling capacitance, total capacitance, and Elmore delay.

To use the standalone reducer, use the StarReduce command at the operating system prompt. The command takes the name of a reducer command file as an argument. For example:

```
%StarReduce red_cmd
```

The valid commands in a reducer command file are explained in the following sections:

- Configuring Results of Reducer Command File
- Specifying File Names
- Specifying an Output Format
- Using the REDUCTION_NETS Command
- Grounding Coupling Capacitors
- Generating a Touchstone File From an SPF File With Parasitic RLCs

Configuring Results of Reducer Command File

This topic lists commands that you can use to configure the output of the reducer command file.

- `COMPRESS_OUTPUT: YES | NO`

  The `COMPRESS_OUTPUT: YES` command generates the output of a netlist in a compressed file even if you specify an input netlist either in an ASCII or a compressed file. The default is `NO`.

- `MERGE_DEVICES: PARALLEL | NO`

  The `MERGE_DEVICES: PARALLEL` command enables merging of parallel devices. The command works only with the SPF or STAR format.

  In a set of parallel devices, the device instance without the at (@) delimiter is the primary device instance. The device instances with the @ delimiter, which follow the primary device instance, are the secondary device instances.

Feedback

When you use the MERGE_DEVICES command, the terminal node names of secondary device instances are changed to have the same name as the primary device instance. This might lead to shorts parasitics between device placements.

- MERGE_INACTIVE_DEVICES: YES | NO

  Inactive devices indicate that the terminals of the device is connected to the same net.

  The MERGE_INACTIVE_DEVICES: YES command allows to merge inactive devices.

- NETLIST_NODE_SECTION: YES | NO

  The NETLIST_NODE_SECTION: YES command controls the subnodes output (*|S or *N) in the reducer output file. The default is YES. When the command is set to NO, the tool reduces the size of the netlist file and it is beneficial in most post-extraction flows.

- REDUCER_NETLIST_COMPRESS: YES | NO

  The REDUCER_NETLIST_COMPRESS: YES command compresses the output of the netlist file.

- REMOVE_NETS: *net_names*

  You need to specify a space-delimited list of nets to remove the specified nets. All coupling capacitors of the nets that are removed are grounded. The * and ? wildcards are acceptable. The default is * (all nets). Case sensitivity of net names is determined by the CASE_SENSITIVE command.

- REMOVE_DEVICES: *model_names*

  You need to specify model names to remove all instances of the devices with the matching model names. It also supports a list of model names separated by a space. The instance ports are converted into sub-nodes and reduced further according to the reduction settings.

- REMOVE_INACTIVE_DEVICES: YES | NO

  Inactive devices indicate that all the terminals of the device is connected to the same net.

  The StarRC tool automatically finds the devices with the REMOVE_INACTIVE_DEVICES: YES command:

  - If the device is connected to one net, the tool removes the device.
  - If the device is connected to two or more nets and there is more than one terminal in the net, the tool makes connections of the devices on that net as ideal by connecting the nets to the top-level port.

Feedback

- OUTPUT_MAX_LINE_LENGTH: *val*

  The actual line length can be longer or shorter than the value specified with the OUTPUT_MAX_LINE_LENGTH command to preserve the words on the line. This means that the tool does not split any words between successive lines. *val* is the number of characters per string. The minimum value is 5.

  The tool breaks the word into multiple lines if a string (word) is longer than the specified value. For example, if the original string is *This is a new file command*, the OUTPUT_MAX_LINE_LENGTH: 10 command returns the output as follows:

```
This is a
new file
command.
```

Specifying File Names

You can specify an input, an output, or a summary file with absolute or relative paths to indicate the location of the file. This topic lists the *_FILENAME commands that you can use in the reducer command file.

- INPUT_FILENAME: *input_name*

  Specify a single file name or GPD directory name.

- OUTPUT_FILENAME: *output_name*

  Specify a file name for the generated file or GPD directory name.

- SUMMARY_FILENAME: *summary_file*

  Specify a file name for the summary file. The default file name is reducer.sum. The tool uses the default file name If you do not specify a summary file name.

  The summary file reports runtime, memory usage by stages, and overall memory usage, including levels of reduction and warning messages, as shown in the following example:

```
Input:  Nets=405   Nodes=40985   R=40195   C=95361   L=0

Netlist parsing  Elp=2.1 Usr=1.46 Sys=0.12 Mem=164.75

Reduction        Elp=2.59 Usr=2.49 Sys=0.03 Mem=188.688

Output: Nets=435   Nodes=6747   R=8447   C=19293   L=0
        FixedNodes=2337

Netlist output   Elp=0.29 Usr=0.26 Sys=0.02 Mem=190.863
```

Feedback

Specifying an Output Format

To specify the output format of the standalone reducer file, use the following command in the reducer command file:

```
OUTPUT_FORMAT: SPF, SPEF or GPD
```

The standalone reducer output file formats are based on the following conditions:

- If the input file is an SPF file, the output format can only be SPF.
- If the input file is an SPEF file, the output format can only be SPEF.
- If the input file is a GPD directory, the output format can be SPF, SPEF, or GPD.
- If the command in the reducer command file is not set, the output format is the same as the input file format.

Using the REDUCTION_NETS Command

The `REDUCTION_NETS` command allows you to specify a space-delimited list of nets to reduce. The * and ? wildcards are acceptable. The default is `*` (all nets). Case sensitivity of net names is determined with the `CASE_SENSITIVE` command.

```
REDUCTION_NETS: net_names
```

When you use the `REDUCTION_NETS: *` or `REDUCTION_NETS: net_names` command, `LEVEL:HIGH` reduction is applied to the specified nets by default.

If you want to avoid applying `LEVEL:HIGH` on certain nets, use the `REDUCTION_NETS: net_names LEVEL NO` command. The following examples show how to use the `REDUCTION_NETS` command with the `LEVEL HIGH` or `LEVEL NO` keyword:

- **`REDUCTION_NETS: * LEVEL HIGH`**

  Applies reduction of level high to all nets with the `MAX_DELAY_ERROR` default 1e-13.

- **`REDUCTION_NETS: A B LEVEL HIGH MAX_DELAY_ERROR 1e-14`**

  Applies reduction of level high to net A and net B with the maximum change allowed in timing delay error is 1e-14.

- **`REDUCTION_NETS: C LEVEL HIGH MAX_DELAY_ERROR 1e-12`**

  Applies reduction of level high only to net C with the maximum change allowed in timing delay error is 1e-12.

- **`REDUCTION_NETS: D LEVEL NO`**

  Applies reduction of level no only to net D.

Feedback

If there are multiple nets to reduce, use the command as shown in the following example to specify each of the nets in the StarReduce command file:

```
REDUCTION_NETS: <net_name1>
REDUCTION_NETS: <net_name2>
…
```

The REDUCTION_NETS: *net_names* command accepts the following optional keywords, which must appear on the same line:

- CASE_SENSITIVE: YES | NO

  Specifies whether net names in the REDUCTION_NETS command are case-sensitive. The default is NO.

- FREQUENCY *freq*

  The frequency at which to calculate the timing. You can supply a frequency instead of the MAX_DELAY_ERROR option. One of the two commands must be present.

- LEVEL HIGH | LEVEL NO

  LEVEL NO does not apply any reduction. The default is LEVEL HIGH.

  The command preserves the parasitic resistor temperature coefficients TC1 and TC2 if present in the input file with LEVEL NO and recalculates the temperature coefficients with LEVEL HIGH.

  REDUCTION_NETS command removes all resistors and nodes, including instance ports and top-level ports of nets that are set with the IDEAL parameter. All couplings are assigned to the node that matches with the net name

- MAX_DELAY_ERROR *timing_err*

  This is the maximum change in timing allowed as a result of the reduction. The default is 1e-13 s.

- MERGE_SHORTED_PORTS: YES | NO

  If the MERGE_SHORTED_PORTS command is set to YES, whenever multiple port nodes for a net are connected together by node-sharing shorting resistors, the standalone reducer chooses one node randomly from the group to represent all nodes. The reducer uses this node to replace every node in the group for every electrical element in the netlist including parasitic elements, elements in the instance section, and *|I occurrences in DSPF. The default is NO.

- REMOVE_DANGLING_BRANCHES

Feedback

The argument removes dangling branches for the specified nets when used with the REDUCTION_NETS command as follows:

```
REDUCTION_NETS: ABC* LEVEL NO REMOVE_DANGLING_BRANCHES
```

If nets are set with the LEVEL NO keyword and not set with the REMOVE_DANGLING_BRANCHES argument, the standalone reducer preserves couplings of the nets from dangling branches. For example, coupling between instances (cells) and dangling branches on shield nets.

Note:

The REMOVE_DANGLING_BRANCHES argument can be used only with the LEVEL NO keyword. The standalone reducer already removes dangling branches with the LEVEL HIGH keyword.

- IDEAL

  Retains capacitances and removes all resistors and nodes, including instance ports and top-level ports of nets that are set with the IDEAL parameter. All couplings are assigned to the node that matches with the net name.

- RONLY

  Removes all coupling and ground capacitances for the specified nets. Coupling capacitances to specified nets from other nets are grounded. Subsequently, the tool applies reduction on all resistors based on the LEVEL option.

- RONLY_KEEPC

  Adds up all coupling and ground capacitances to the *|P node for the specified nets. Subsequently, the tool applies reduction on all resistors based on the LEVEL option.

- SHORT_RES *<value>*

  Shorts all resistors in the netlist with resistance less than the specified value. Specify the value in ohms. The default is 0, so the REDUCTION_NETS command does not short any resistors. For example,

  ```
  REDUCTION_NETS: LEVEL HIGH SHORT_RES 1000 LUMP_CAP 1e-17
  ```

- SHORT_RES_LAYER *<layer>*

  Specify the LAYER parameter If there are layer and PIC level maps in the netlist, so the command shorts the resistors only present above the specified layer. For example,

  ```
  REDUCTION_NETS: LEVEL HIGH SHORT_RES 50 SHORT_RES_LAYER METAL2
   LUMP_CAP 1e-15
  ```

  To see the PIC level of a map in the netlist, see NETLIST_PIC_LEVEL_MAP.

Feedback

- LUMP_CAP *<value>*

  Lumps all capacitors to an instance port or pin less than the specified value. The capacitors are added to the first subnode if there are no instance ports or pins. Specify the value in farads. The default is 0, so the REDUCTION_NETS command does not lump any capacitors.

Examples of Reducer Application

This topic lists examples of reducer application.

Lumping Capacitors

The following examples show how the REDUCTION_NETS command lumps capacitors:

- **REDUCTION_NETS: LEVEL HIGH SHORT_RES 1e16**

  Shorts all resistors, and retains all coupling capacitors.

- **REDUCTION_NETS: LEVEL HIGH SHORT_RES 1e16 LUMP_CAP 1**

  Shorts all resistors, and lumps all capacitors.

- **REDUCTION_NETS: LEVEL HIGH SHORT_RES 1e16 LUMP_CAP 1e-17**

  Shorts all resistors, and lumps coupling capacitors that are below 1e-17.

- **REDUCTION_NETS: LEVEL HIGH SHORT_RES 1000 LUMP_CAP 1e-17**

  Shorts resistors that are less than 1000 and lumps coupling capacitors that are below 1e-17.

- **REDUCTION_NETS: LEVEL HIGH SHORT_RES 50 SHORT_RES_LAYER METAL2 LUMP_CAP 1e-15**

  Shorts resistors that are less than 50 and lumps coupling capacitors that are below 1e-15.

Using the LEVEL Parameter

The following examples show how to use the LEVEL parameter to apply reduction:

- **REDUCTION_NETS: * LEVEL_NO REMOVE_DANGLING_BRANCHES**

  Removes dangling branches without reduction.

- **REDUCTION_NETS: * LEVEL HIGH MERGE_SHORTED_PORTS**

  Removes node sharing resistors from a netlist file, might impact back-annotation flow.

- **REDUCTION_NETS: * LEVEL HIGH LUMP_CAP** *value*

Feedback

Merges coupling capacitances and merging is not based on MAX_DELAY_ERROR.

- **REDUCTION_NETS: * LEVEL HIGH MAX_DELAY_ERROR IDEAL**

  Lists nets that are set with the IDEAL parameter as ideal nets in the output file.

- **REDUCTION_NETS: abc* LEVEL HIGH MAX_DELAY_ERROR RONLY**

  Removes all coupling and ground capacitances for the specified nets. Coupling capacitances to specified nets from other nets are grounded. Subsequently, the tool applies reduction on all resistors based on the LEVEL option.

- **REDUCTION_NETS: def* LEVEL HIGH MAX_DELAY_ERROR RONLY_KEEPC**

  Adds up all coupling and ground capacitances to the *|P node for the specified nets. Subsequently, the tool applies reduction on all resistors based on the LEVEL option.

Grounding Coupling Capacitors

This topic lists commands to use in the reducer command file that helps you to ground coupling capacitors.

- COUPLING_ABS_THRESHOLD: *value*

  Specify the absolute threshold value to ground coupling capacitors. The default is 0, so the StarReduce command does not ground any coupling capacitors.

- COUPLING_REL_THRESHOLD: *value*

  Specify the relative threshold value to ground coupling capacitors. The value can be between 0 and 1. The default is 0, so the StarReducer command does not ground any coupling capacitors.

- COUPLING_THRESHOLD_OPERATION: AND | OR

  Filters coupling capacitance. The default is AND. Before using this command, you must specify the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands.

Figure 103 Condition 1

Coupling capacitance of net1 and net2 is less than COUPLING_ABS_THRESHOLD

Figure 104 Condition 2

Coupling capacitance of net1 and net2 is less than COUPLING_REL_THRESHOLD * TCAP_net1

Feedback

Figure 105 Condition 3

Coupling capacitance of net1 and net2 is less than `COUPLING_REL_THRESHOLD * TCAP_net2`

When you specify the following filters with the `COUPLING_THRESHOLD_OPERATION` command,

- `COUPLING_THRESHOLD_OPERATION: AND`

  Decouples coupling capacitance if the Example 13 and (Example 13 and Example 13) conditions for coupling operation are true.

- `COUPLING_THRESHOLD_OPERATION: OR`

  Decouples coupling capacitance if the Example 13 or (Example 13 and Example 13) condition for coupling operation is true.

Generating a Touchstone File From an SPF File With Parasitic RLCs

The following commands generate a touchstone file based on group nets selected from the input SPF file with parasitic resistance, inductance, and capacitance (RLC). The commands convert the group nets from the input SPF file into S parameters. The output SPF file has the full RLC information with S-element models for the selected nets (converted to S parameters). The input SPF file might be generated during clock net inductance extraction.

- `S_PARAM_NETS: sparam_output_file net_names`

  Specify an output file in the Touchstone format and a space-delimited list of nets for S-parameter analysis.

- `S_PARAM_FREQUENCY: (LIN | DEC) npoints start_freq stop_freq`

  Specify a frequency range for S-parameter calculation. For descriptions of the arguments, see `INDUCTANCE_S_PARAM_FREQUENCY`.

You can also define two groups of nets or more than two groups of nets to calculate corresponding S parameters and produce the corresponding Touchstone files, as shown in Example 14. Also, the commands create a SPF output file with the following information:

- The original nets that are not converted into S parameters.
- The S-element model replaces the selected nets (converted to S parameters) and their information is represented in the Touchstone file.

Feedback

Example 14 Defining Groups of Nets Using S-Parameter Commands

```
INPUT_FILENAME: input_SPF_file.spf

S_PARAM_NETS: grp1_xyz_starreduce_2NETs.31.30_10ports gab_rx_data_0[31]
 gab_rx_data_0[30] S_PARAM_FREQUENCY: LIN 3 0.1G 10G
S_PARAM_NETS: grp2_xyz_starreduce_3NETs.29.28.27_12ports
 gab_rx_data_0[29] gab_rx_data_0[28] gab_rx_data_0[27] S_PARAM_FREQUENCY:
 LIN 3 0.1G 10G
```

Example 14 shows that the two groups are specified: group1 with 2 nets (each net with 5 ports) and group 2 with 3 nets (net1 with 5 ports, net3 with 5-ports, and net3 with 2 ports).

Feedback

Feedthrough Nets

A feedthrough net is a net that has one connection to a higher level in the hierarchy. Feedthroughs do not typically exist in a schematic. The StarRC tool handles the following feedthrough cases:

- Extracting lower-level cells that are later used as `SKIP_CELLS`, as shown in Figure 106.
- Ensuring proper naming for the feedthrough ports of a skip cell. Figure 107 shows an example of extracting the TOP block by skipping the cell with feedthrough ports.

Both issues should be taken care of by default in `XREF:YES` because it is layout-based and the feedthrough exists in the layout.

Figure 106 First Feedthrough Case

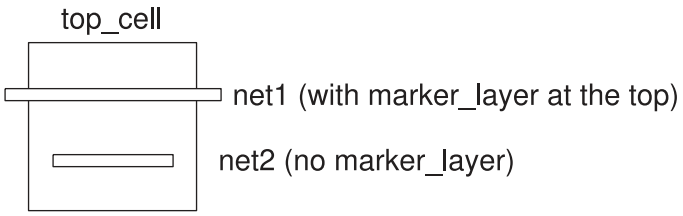
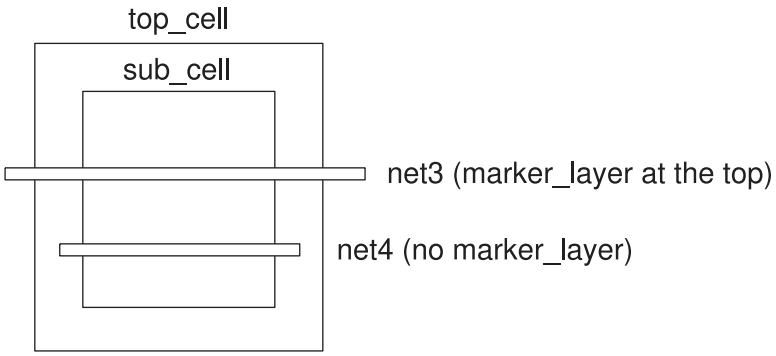


Figure 107 Second Feedthrough Case



To handle these cases with `XREF:COMPLETE`, use the `XREF_FEEDTHRU_NETS` command, for which the default is `NO`. This command controls feedthrough net handling for the two previous cases when `XREF:COMPLETE` is used.

Feedback

Note:

Both of these issues can happen for a single cell. For example, if a cell has a SKIP_CELLS with feedthrough ports and has a feedthrough port for a bigger cell containing the cell itself, both *|I and *|P are correctly treated.

Extracting Lower-Level Feedthrough Cells

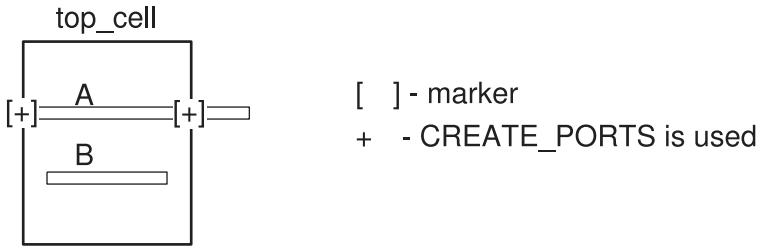
The first feedthrough problem is cross-referencing *|P for the feedthrough ports of a subblock when extracting the subblock.

If nets or ports at the top cell of the layout are not compared, the netlist of those nets appears with the “ln_” prefix attached to the net names from the layout. This is done by default for XREF:YES, but for XREF:COMPLETE, the XREF_FEEDTHRU_NETS command must be set to YES (the default is NO). The prefix itself can be changed with the XREF_LAYOUT_NET_PREFIX command.

The XREF:COMPLETE flow always prints out the prefix for the feedthrough net name, because the net name is always based on the layout.

In the Hercules runset, the CREATE_PORTS command must be used for feedthrough ports in the XREF:COMPLETE flow and included in the netlist.

Figure 108 XREF:YES Example



The output is:

```
…
*|NET ln_A
*|P ln_A
…
*|NET ln_B
…
```

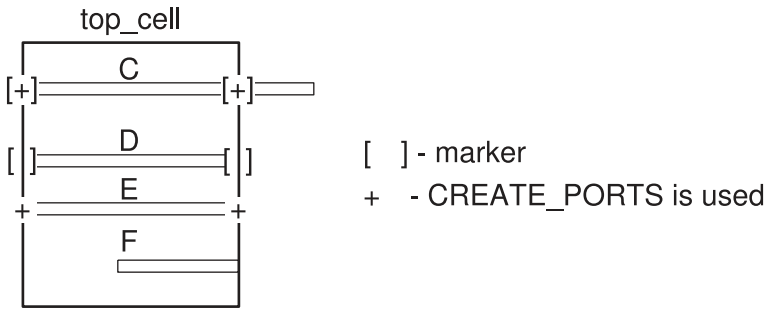
Feedback

Port Renaming

To rename feedthrough ports, you can use the SHORT_PINS:NO option. For example, if SHORT_PINS:NO, the output would be

```
…
*|NET ln_A
*|P ln_A_1
*|P ln_A_2
…
*|NET ln_B
```

Figure 109 XREF:COMPLETE and XREF_FEEDTHRU_NETS:YES Example



The output is:

```
…
*|NET ln_C
*|P ln_C
…
*|NET ln_E
…
```

Note that nets D and F are not written in the output netlist, because the CREATE_PORTS command (Hercules) was not used.

Naming Feedthrough Ports

When you are cross-referencing *|I for skip cell feedthrough ports (when extracting the top block with skip cells that have a feedthrough, as in Figure 107), the XREF:COMPLETE command has the same behavior as the XREF:YES command if the XREF_FEEDTHRU_NETS command is set to YES, even though the schematic does not have the feedthrough port connection with the skip cells.

When the XREF_FEEDTHRU_NETS command is set to NO and the XREF command is set to COMPLETE, the StarRC tool ignores the *|I and the instance section also skips the connection.

Feedback

If a SPICE subcircuit file is provided for the skip cells (by using the `SPICE_SUBCKT_FILE` command), then the order and content from the SPICE subcircuit file is maintained in the instance section of the netlist.

Runset Requirements

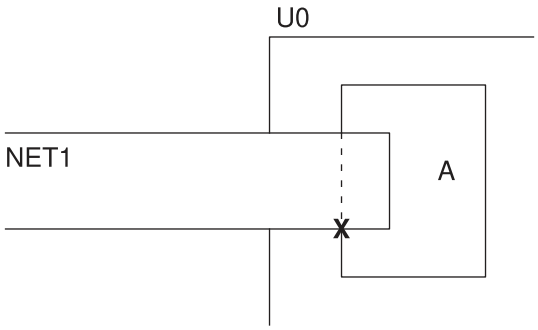
- Hercules LVS for the skip cells that have the feedthrough ports must be done with EQUIV statements. Using the Hercules command `BLACK_BOX` in the equiv file is not permitted.
- The Hercules `CREATE_PORTS` command must be used to properly netlist feedthroughs.

Long Ports

This feature is used to force *|I reported port locations in both top-level and cell-level (for `INSTANCE_PORT:NOT_CONDUCTIVE`) coordinate systems to the coordinates of the physical overlap of the top-level route with the instance port shape.

In the simplest case, the xy coordinate written for the *|I port is the lower-left corner of the overlapping region. These regions are called PortUpContacts. See Figure 110.

Figure 110 PortUpContact Example



The X in Figure 110 represents the intersection coordinate (PortUpContact node) for the NET1 to U0/A connection. For longer overlay-type connections, multiple PortUpContact nodes are extracted. These nodes are located in the same way that subnodes on a net are extracted. For multiple separated connections of a single net to a single port (multiple intersection), multiple PortUpContact nodes are extracted, one for each overlapping area.

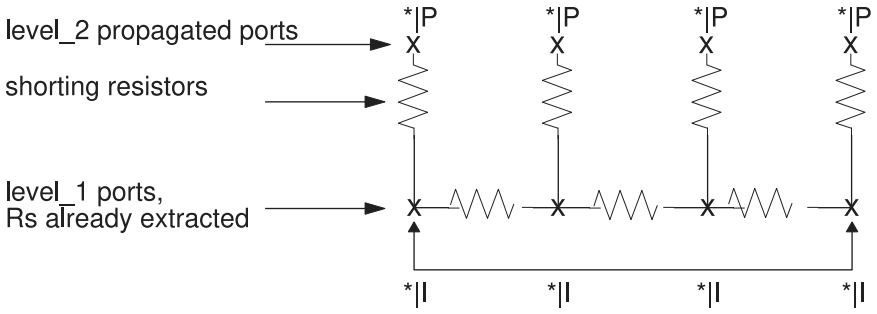
When a propagated port is extracted, it is shorted to the lower level by a shorting resistor, but resistance extraction does not occur at that level. It is assumed that the extraction has already occurred at the lower level.

In Figure 111, the level_2 propagated port is written with multiple *|P statements, using global coordinates. The level_1 ports are written with multiple *|I statements, using

Feedback

local coordinates for the level_1 instance, and they are shorted to level_2 with shorting resistors.

Figure 111 Propagated Ports



Feedback

Via Coverage

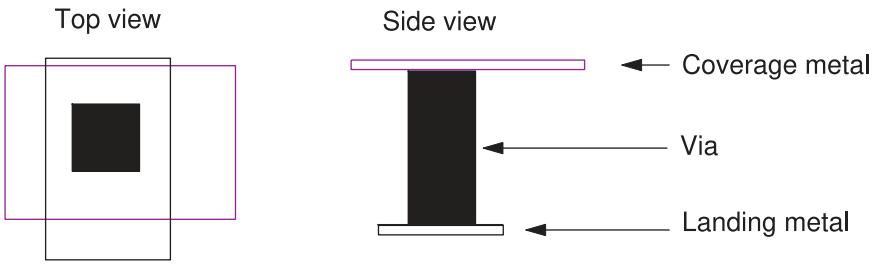
The StarRC tool can analyze via coverage, which is the amount by which a metal polygon that connects to a via overlaps the via polygon. Via coverage is important for reliability analysis and is analyzed separately for the top and bottom metal polygons.

Figure 112 illustrates the relationship between the polygons of interest. The top-level metal polygon is referred to as the coverage layer (polygon) and the bottom-level metal is the landing layer (polygon).

Note:

The term via coverage refers to the overall via coverage analysis technique. The terms coverage layer, coverage polygon, and coverage metal refer to the specific metal polygon in the metal layer above the via.

Figure 112 Coverage and Landing Metal With Respect to a Via



Via coverage analysis works as follows:

1. The StarRC tool analyzes the layout of the via, coverage metal, and landing metal polygons and generates a set of geometric parameters that describe the layout.
2. You define a set of criteria against which to test the via geometric parameters.
3. The tool uses the test criteria to evaluate the via geometric parameters.
4. The via is classified into a category, which is reported in the netlist tail comments as the $vc parameter. A reference table in the netlist lists the category codes.

You can enable via coverage analysis and define the coverage parameters in either of the following ways:

- Specify the `VIA_COVERAGE` command in the ITF file (nxtgrd file).
- Use one or both of the `VIA_COVERAGE` and `VIA_COVERAGE_OPTION_FILE` commands in the StarRC command file.

Feedback

If the ITF file contains a VIA_COVERAGE command and the StarRC command file contains either the VIA_COVERAGE or VIA_COVERAGE_OPTION_FILE command, the tool issues an error message.

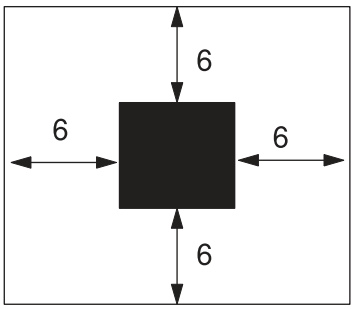
Determining the Coverage and Landing Areas for Rectangular Vias

Use the VIA_COVERAGE_OPTION_FILE command to specify a file containing checking rules for rectangular vias. Alternatively, you can use the VIA_COVERAGE command in the ITF file (nxtgrd file).

The following conditions determine via coverage and landing:

- If all edges are enclosed by the full- coverage parameter, the via is fully covered. See Figure 113.

Figure 113 Via Rules for Verifying Full Coverage



If all sides have coverage and landing metal coverage greater than the full (F) enclosure parameter, the via is fully enclosed.

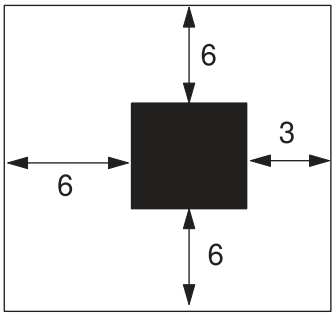
F = 5
Q2 = 4
Q1 = 1
S2 = 2
S1 = 1

Example via is fully covered because all enclosures are greater than the F parameter.

- If the via is not fully covered, it might be quarter covered. If one edge has enclosure greater than or equal to Q1, and BOTH adjacent edges have enclosure greater than Q2, the via is quarter covered. The opposite edge must also have an enclosure greater than or equal to Q1. See Figure 114.

Feedback

Figure 114 Via Rules for Verifying Quarter Coverage



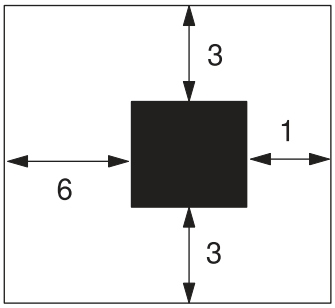
For quarter coverage, one enclosure must be greater than or equal to Q1. Must have both adjacent sides enclosed by more than Q2.

F = 5
Q2 = 4
Q1 = 1
S2 = 2
S1 = 1

Example via is quarter covered because one enclosure has greater than Q1 (3m) and adjacent edges have an enclosure greater than Q2.

- If the via is not quarter covered, it might be semicovered. If one edge has enclosure greater than or equal to S1, and BOTH adjacent edges have enclosure greater than S2, the via is semicovered. The opposite edge must also have enclosure greater than or equal to S1. See Figure 115.

Figure 115 Via Rules for Verifying Semi Coverage



For semicoverage, one enclosure must be greater than or equal to S1. Both adjacent sides must be enclosed by more than S2.

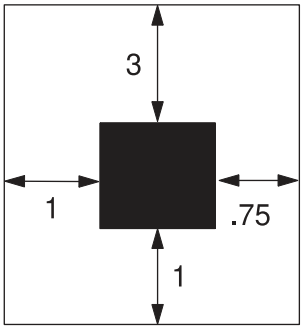
F = 5
Q2 = 4
Q1 = 1
S2 = 2
S1 = 1

Example via is semicovered because one edge has an enclosure greater than or equal to S1 (1m), both adjacent edges have an enclosure greater than S2, and adjacent sides are enclosed by less than Q2.

- If none of the preceding conditions is met, the via is partially covered as shown in Figure 116.

Feedback

Figure 116 Via Rules for Verifying Partial Coverage



For partial coverage, no enclosure meets the conditions of full, quarter, or semicovered.

F = 5
Q2 = 4
Q1 = 1
S2 = 2
S1 = 1

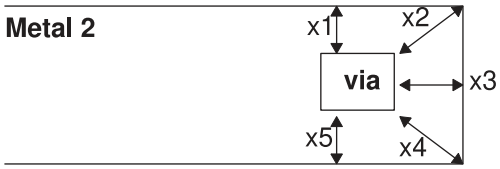
Example via is partially covered because no edge meets the full, quarter, or semicovered requirements

- In instances where a via appears to satisfy both the quarter coverage and semicoverage, the via is considered to be quarter covered.

Determining the Coverage and Landing Areas for Square Vias

Use the `VIA_COVERAGE` command to define the via coverage and landing areas for square vias. Alternatively, you can use the `VIA_COVERAGE` command in the ITF file (nxtgrd file).

Figure 117 VIA_COVERAGE Behavior



Coverage for the via in Figure 117 is determined by examining the minimum distance, considering the values x1 through x5. Table 48 shows the definitions.

Table 48 Via Coverage Definitions

Minimum Distance	Coverage
Greater than or equal to full coverage	(F) Fully covered
Greater than or equal to quarter coverage and less than full coverage	(Q) Quarter covered
Greater than or equal to semi coverage and less than quarter coverage	(S) Semi covered

Feedback

Table 48 Via Coverage Definitions (Continued)

Minimum Distance	Coverage
Less than semi coverage	(P) Partially covered

Positive and Negative Check

Positive and negative checks using `VIA_COVERAGE` are described in the following section. This concept is shown in Figure 118.

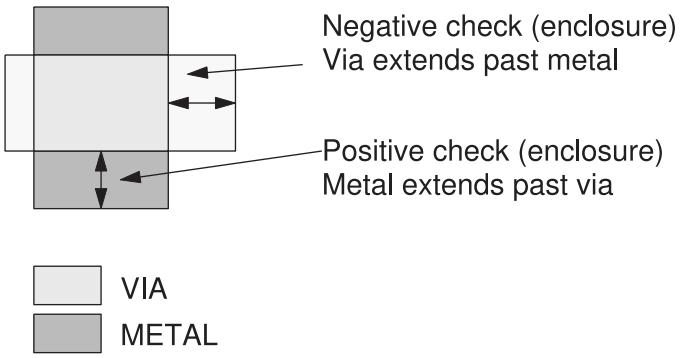
- Positive check

  Metal edges extend beyond the via edges and metal encloses the via.

- Negative check

  Via extends beyond the edges of the metal polygons.

Figure 118 Positive and Negative Checks



Of the two commands supporting via coverage capabilities, `VIA_COVERAGE` and `VIA_COVERAGE_OPTION_FILE`, the negative check is only supported in the `VIA_COVERAGE_OPTION_FILE` command.

For the negative check, you specify negative parameters in the `VIA_COVERAGE_OPTION_FILE` command. For metal parameters, the StarRC tool performs the negative check with greater than or equal to values of the negative parameter. If you specify a negative value in a `VIA_COVERAGE` command, the tool issues an error message. The values of check parameters you specify in the `VIA_COVERAGE_OPTION_FILE` command can be zero. Any coverage larger or equal to zero (for example, nonnegative) satisfies the zero coverage and landing check.

Feedback

Examples

The examples in this section describe the negative check. Table 49 defines the F1, F2, Q1, Q2, S1, and S2 parameters.

Table 49 Negative Check

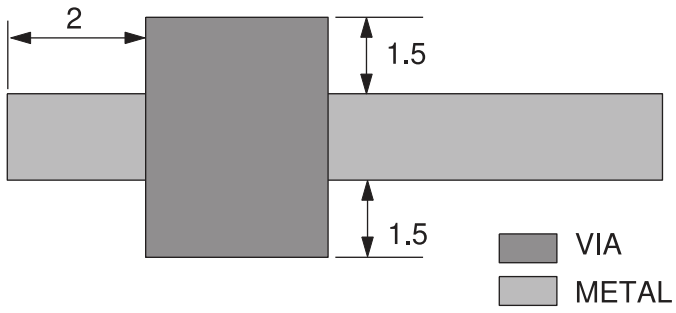
Parameter	Definition
F1	First landing or coverage parameter for a full check
F2	Second landing or coverage parameter for a full check
Q1	First landing or coverage parameter for a quarter check
Q2	Second landing or coverage parameter for a quarter check
S1	First first landing or coverage parameter for a semi check
S2	Second landing or coverage parameter for a semi check

Example 1

- Case 1: Q1=-1 and Q2=1

  The geometries do not meet Q1= -1 and Q2 = 1 because via edges extend beyond metal by 1.5, which is more than 1. Via coverage equals -1.5 which is smaller than -1. This is shown in Figure 119.

Figure 119 Example 1, Case 1

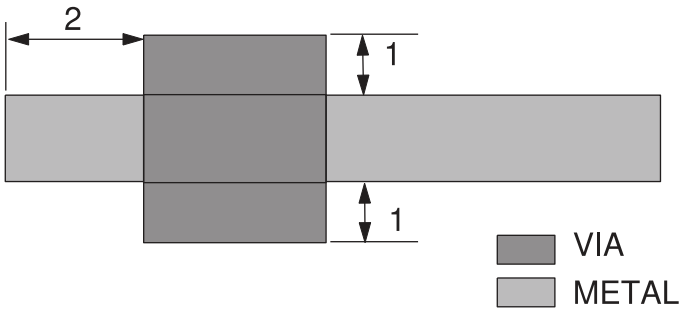


- Case 2: F2= 5, F1= -1; Q2= 3, Q1= -1; S2= 3, S1= -2

Feedback

The geometries meet Q1= -1 and Q2=1. Because via edges extend beyond the metal edge by 1, satisfying Q1= -1; and the other adjacent opposite metal edges enclose the via by distances larger or equal to 2. This is shown in Figure 120.

Figure 120 Example 1, Case 2

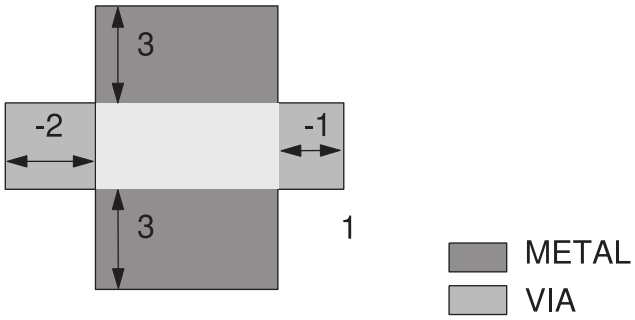


Example 2

The parameters are F2=5, F1=-1; Q2=3, Q1=-1; S2=3, S1=-2. These are shown in Figure 121.

- The geometries do not meet F2=5 and F1=-1 because via extends past metal by more than 1 and the adjacent enclosures are less than 5.
- The geometries do not meet Q2=3 and Q1-1 because one via edge extension past metal is -2, which is less than -1, although another opposite enclosure is equal to -1 and the adjacent enclosures are equal to 3.
- The geometries meet S2=3 and S1=-2 because the via extension past metal is greater than or equal to -2 and the adjacent enclosures are equal to 3.

Figure 121 Example 2



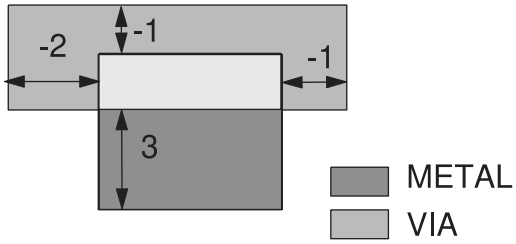
Feedback

Example 3

The parameters are F2=5, F1=-1; Q2=3, Q1=-1, S2=1, S1=-1, as shown in Figure 122.

- The geometries do not meet F2=5 and F1=-1 because no two opposite enclosures areas are greater than or equal to 5.
- The geometries do not meet Q2=3 and Q1=-1 because no two opposite enclosures are greater than or equal to 3.
- The geometries do not meet S2=1 and S2=-1 because no two opposite enclosures have area greater than or equal to 1.

Figure 122 Example 3



Output

The via coverage output for a negative check does not have an impact on the output format.

Via Coverage Examples

The `VIA_COVERAGE` command supports the checking of square vias and the `VIA_COVERAGE_OPTION_FILE` command supports the checking of rectangular vias.

VIA_COVERAGE Syntax With Semicoverage

You can specify the `VIA_COVERAGE` command with the semicoverage function as follows.

```
VIA_COVERAGE: via_layer_name Lf Lq [Ls] Cf Cq [Cs]
```

```
NETLIST_FORMAT: SPEF
NETLIST_TAIL_COMMENTS: YES
VIA_COVERAGE: VIA1 100 80 40 100 80 40
VIA_COVERAGE: VIA2 100 80 40 100 80 40
```

Feedback

VIA_COVERAGE Syntax Without Semicoverage

For backward compatibility, you can specify the `VIA_COVERAGE` command without the semicoverage function.

```
VIA_COVERAGE: via_layer_name Lf Lq Cf Cq

NETLIST_FORMAT: SPF
NETLIST_TAIL_COMMENTS: YES
VIA_COVERAGE: VIA1 100 80 100 80
VIA_COVERAGE: VIA2 100 80 100 80
```

VIA_COVERAGE_OPTION_FILE - Syntax With Semicoverage

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing =
FL,QL1,QL2,[SL1,SL2];Coverage=FC,
QC1,QC2,[SC1,SC2]}
(Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;
Landing=FL,QL1,QL2,[SL1,SL2];Coverage=FC,QC1,QC2,[SC1,SC2]}
```

VIA_COVERAGE_OPTION_FILE - Syntax Without Semicoverage

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing =
FL,QL1,QL2;Coverage=FC,QC1,QC2}
(Xrange-Xmin,Xmax;Yrange=Ymin,Ymax;Landing =
FL,QL1,QL2;Coverage=FC,QC1,QC2}
```

Reading the Output Report

The results from the analysis appear under the heading of VIA_COVERAGE_CODES in the netlist and as a comment in the resistor element.

Coordinate locations of the reported vias are found in the SPF and SPEF outputs.

Report Differences

The report looks different depending on which via coverage commands you specify. Commands that include the semicoverage check generate more results.

When Checking Semicoverage

You can read the results of the semicoverage check in the netlist file. Reading the lines horizontally, find the specific check in the index line. The first letter indicates the via landing and the second letter indicates the via cover. As shown in Example 15, the SS column indicates semivia landing and semivia coverage. Similarly, PP indicates partial via landing and partial via coverage and so on.

Feedback

Example 15 Semicoverage Codes for a Square or Rectangular Check With Semi Results

```
// *index FF FQ FS FP QF QQ QS QP SF SQ SS SP PF PQ PS PP X_by_Y
// *       'FQ' stands for Full landing and Quarter coverage
// *0     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
// *1     0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1x1
// *2     0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1x1
// *3     0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1x1
// *4     0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1x1
// *5     0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1x1
// *6     0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1x1
// *7     0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1x1
// *8     0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1x1
// *9     0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1x1
// *10    0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1x1
// *11    0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1x1
// *12    0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1x1
// *13    0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1x1
// *14    0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1x1
// *15    0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1x1
// *16    1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1x1
```

The via coverage code shown in Example 15 is for the following net:

```
*|NET FF 0PF
*|S (FF:1 0.05 0.05) // $llx=-0.1 $lly=-0.1 $urx=0.2 $ury=0.2 $lvl=1
*|S (FF:2 0.05 0.05) // $llx=-0.1 $lly=-0.1 $urx=0.2 $ury=0.2 $lvl=2
R16 FF:1 FF:2 23.67 $vc=16 $a=0.01 $lvl=3
```

The coverage code table output when semi is not checked is similar, except the columns indicating semi via landing or semi via coverage are not included, as shown in Example 16.

Example 16 Semicoverage Codes for a Square or Rectangular Check With Semi Results

```
// *index FF FQ FP QF QQ QP PF PQ PP X_by_Y
// *       'FQ' stands for Full landing and Quarter coverage
// *0     0  0  0  0  0  0  0  0  0  0x0
// *1     0  0  0  0  0  0  0  0  1  1x1
// *2     0  0  0  0  0  1  0  0  0  1x1
// *3     0  0  1  0  0  0  0  1  0  1x1
// *4     0  0  0  0  0  0  0  1  0  1x1
// *5     0  0  0  0  1  0  0  0  0  1x1
// *6     0  1  0  0  0  0  0  0  0  1x1
// *7     0  0  0  0  0  0  1  0  0  1x1
// *8     0  0  0  1  0  0  0  0  0  1x1
// *9     1  0  0  0  0  0  0  0  0  1x1
```

The via coverage code shown in Example 16 is for the following example net:

```
*|NET PP 0PF
*|S (PP:1 1.85 -1.45) // $llx=1.78 $lly=-1.52 $urx=1.92 $ury=-1.38 $lvl=1
*|S (PP:2 1.85 -1.45) // $llx=1.78 $lly=-1.52 $urx=1.92 $ury=-1.38 $lvl=2
R1 PP:1 PP:2 23.67 $vc=1 $a=0.01 $lvl=3
```

Feedback

In an SPF file, a parameter is added to the tail comment: $vc. This is an integer that identifies the coverage code. The key for the codes is located in a file named via_coverage_codes located in the star directory.

The X_by_Y column is written for via arrays. In Example 16, the two resistors in the netlist are the same 2-by-5 via array, with the last one rotated by 90-degrees.

Net0 (noncritical) polygons are not used in the via coverage calculation and should not be considered. All relevant neighbor polygons are considered when the context of the VIA is being analyzed.

Part 2: Process Modeling

11

Process Modeling Methodology

The nxtgrd file describes the relationship between the design layout and the manufacturing process, which is essential information for calculating accurate parasitics. For signoff flows, you must obtain an nxtgrd file from your foundry. For process exploration, you can create an nxtgrd file using the provided grdgenxo tool.

For more information, see the following topics:

- Flows for Process Characterization
- The Interconnect Technology Format File
- The Mapping File
- The grdgenxo Command

Feedback

Flows for Process Characterization

Determining the capacitance and resistance of a circuit requires detailed modeling of the process technology. The StarRC tool uses pattern matching to analyze a design by comparing the layout to a set of primitive structures whose parasitics have been previously analyzed. Those reference parasitics are contained in a process characterization database called the New Xtraction Generic Regression Database or nxtgrd file.

Caution:

For signoff flows, you must use the nxtgrd files provided by your foundry. Foundries develop nxtgrd files for their processes and distribute them in encrypted form.

The nxtgrd file also contains information for the TLUPlus model, which is a simplified model for approximating process effects. During parasitic extraction, the StarRC tool uses the detailed process information. Place-and-route or synthesis tools use the TLUPlus information in early design stages.

The nxtgrd file is created by the grdgenxo tool, a StarRC utility program, based on information in an ITF file. An ITF file describes the fabrication process using a process description language called the Interconnect Technology Format (ITF). The terms ITF file and nxtgrd file are sometimes used interchangeably; however, the ITF file is the input (the process description) and the nxtgrd file is the output (the file used during StarRC extraction).

You can use ITF files in several ways, as described in the following sections:

- The grdgenxo Flow

  Run the grdgenxo tool to process the ITF file and create the nxtgrd file for use in StarRC extraction (specified in the StarRC command file with the `TCAD_GRD_FILE` command).

- The Direct ITF Flow

  Use the StarRC tool to read the ITF file directly by using the `ITF_FILE` command. The tool uses the field solver to analyze the capacitance of the design. This operation is supported only for capacitance extraction in transistor-level flows.

- The QTF Flow

  For advanced modeling of complex device structures, foundries might provide a process file called the QuickCap Technology File (QTF). The QTF file is the technology file for the QuickCap tool, a standalone capacitance field solver. The StarRC tool uses its built-in field solver to analyze the QTF file for better accuracy in the device region.

Feedback

The grdgenxo Flow

The grdgenxo tool generates the nxtgrd file (the process database file) from an ITF file, as illustrated in Figure 123.

Figure 123 Process Database File Generation

The grdgenxo tool uses an internal field solver operating on an extensive set of primitive structures to generate the nxtgrd file. The nxtgrd file contains capacitance, resistance, and layer information. The ITF statements are also included in the file.

The nxtgrd file is automatically compressed with the gzip utility during file creation. In addition, most of the file content is encrypted. However, header information is not encrypted, such as the grdgenxo version used to create the file and the ITF commands used to model the layers. To view the unencrypted header information without extracting the entire file, use the following command at the operating system prompt:

```
% zless nxtgrd_file
```

The Direct ITF Flow

To experiment with process changes in transistor-level flows, you can specify an ITF file directly by using the `ITF_FILE` command in the StarRC command file. The StarRC tool automatically uses the field solver to analyze the capacitance of the process structures.

The direct ITF flow does not use the grdgenxo tool and does not generate or use the nxtgrd file. This flow is supported only for capacitance extraction in transistor-level flows.

You cannot use extraction results from the direct ITF flow for design signoff, because your process definition might not represent the foundry manufacturing process.

Table 50 lists ITF and StarRC commands that cannot be used with the direct ITF flow.

Feedback

Table 50 Unsupported ITF and StarRC Commands for the Direct ITF Flow

Unsupported ITF Commands	Unsupported StarRC Commands
AREA CRT_VS_AREA CRT_VS_SI_WIDTH CRT1, CRT2, and T0 DEVICE_TYPE DIELECTRIC_FILL_EMULATION_VS_SI_SPACING DROP_FACTOR DROP_FACTOR_LATERAL_SPACING ER_TABLE ETCH_VS_CONTACT_AND_GATE_SPACINGS EXTENSIONMIN all commands with prefix FILL_ all commands with prefix GATE_ GLOBAL_TEMPERATURE RESISTIVE_ONLY_ETCH RHO and all commands with prefix RHO_ RPSQ and all commands with prefix RPSQ_ RPV and all commands with prefix RPV_ SPACER_ER_VS_WIDTH_AND_SPACING TSV USER_DEFINED_DIFFUSION_RESISTANCE	3D_IC EXTRACTION (except with the C option) EXTRA_GEOMETRY_INFO (except with the NONE option) FS_EXTRACT_NETS NETLIST_POWER_FILE TCAD_GRD_FILE TEMPERATURE_SENSITIVITY

Table 51 lists ITF commands that are supported in the direct ITF flow.

Feedback

Table 51 Supported ITF Commands for the Direct ITF Flow

Supported ITF Commands	Supported ITF Commands (continued)
ASSOCIATED_CONDUCTOR	IS_CONFORMAL
BACKGROUND_ER	IS_PLANAR
BOTTOM_DIELECTRIC_ER	LATERAL_CAP_SCALING_VS_SPACING
BOTTOM_DIELECTRIC_THICKNESS	LAYER_TYPE
BOTTOM_THICKNESS_VS_SI_WIDTH	MEASURED_FROM
BW_T	MULTIGATE
CAPACITIVE_ONLY_ETCH	POLYNOMIAL_BASED_THICKNESS_VARIATION
CONDUCTOR	all commands with prefix PROCESS_
DAMAGE_ER	all commands with prefix RAISED_DIFFUSION_
DAMAGE_THICKNESS	REFERENCE_DIRECTION
DENSITY_BOX_WEIGHTING_FACTOR	all commands with prefix SIDE_TANGENT_
DIELECTRIC	SMIN
DIELECTRIC_FILL_VS_SI_SPACING	SW_T
ER	TALL_VIA_CONFIG
ER_VS_SI_SPACING	TECHNOLOGY
ETCH	all commands with prefix THICKNESS_
ETCH_VS_WIDTH_AND_LENGTH	TO
ETCH_VS_WIDTH_AND_SPACING	TW_T
FROM	USE_SI_DENSITY
HALF_NODE_SCALE_FACTOR	VIA
ILD_VS_WIDTH_AND_SPACING	WMIN

Feedback

The QTF Flow

To model complex device structures, foundries might provide an optional process file called the QuickCap Technology File (QTF), usually in encrypted form. The QTF flow is supported only for capacitance extraction in transistor-level flows.

StarRC extraction uses the QTF file only for certain device structures. You must do the following to use the QTF flow:

- Specify the nxtgrd file to model the rest of the process.
- Add the `map_qtf_layers` and `qtf_layers` sections in the mapping file.

QTF files are always processed by the field solver. Runtime for the QTF flow is longer than for standard extraction flows.

The following is a general procedure for a StarRC flow that uses a QTF file. In this flow, the nxtgrd file, QTF file, and mapping file are usually supplied by the foundry.

1. Use the `TCAD_GRD_FILE` command to specify the nxtgrd file.
2. Use the `FS_QTF_FILE` command to specify the QTF file.
3. Use the `MAPPING_FILE` command to specify the mapping file. This mapping file should include the `map_qtf_layers` and `qtf_layers` sections.

   Otherwise, use the `QTF_MAPPING_FILE` command to specify a QTF mapping file, so the tool generates the `map_qtf_layers` and `qtf_layers` sections.
4. (Optional) Use the `FS_QTF_OPTIONS` command to specify the reference direction for directional etches. The default is `None`. To specify the y-direction, use the following command:

   ```
   FS_QTF_OPTIONS: -qtfReference y
   ```
5. Use the `FSCOMPARE_OPTIONS` command to specify field solver options, such as convergence goals and distributed processing conditions.
6. Run the StarRC extraction.

For more information about the QuickCap standalone capacitance field solver, see the QuickCap documentation on SolvNetPlus.

Feedback

The Interconnect Technology Format File

An ITF file defines a cross section profile of the process. The file contains an ordered list of conductor and dielectric layer definition statements. The layers are defined from the topmost dielectric layer to the bottommost dielectric layer, excluding the substrate. The ITF parameters are specified layer by layer in a way that is consistent with the physical process. The lowest layer in the ITF cross section must be a dielectric layer. The `SUBSTRATE` keyword refers to a special conductor whose top plane is at 0 height, underneath the lowest dielectric layer. Do not define `SUBSTRATE` in the ITF file. The via layers are defined relative to valid conducting layers. The heights of the conductors and dielectrics are determined by the order in which they are specified and by the thicknesses of the lower layers. When you are specifying a new conductor or dielectric layer, the bottom plane of that layer is exactly the top plane of the lowest dielectric layer unless a `MEASURED_FROM` statement is included to explicitly specify the location of the bottom plane. The lowest dielectric—the lowest physical layer—listed in the ITF file is automatically measured from the `SUBSTRATE` layer. A fully planar process, in which the process cross section does not contain any vertically intersecting conductors at different heights, is the simplest model. For an example, see Fully Planar Process ITF Example.

Using the ITF file to describe a process technology eliminates the need to check parameter sets for consistency. If the sheet resistance parameters for a process must be altered, it is best to regenerate the nxtgrd file. Regenerating the file with updated sheet resistances is very fast, because the grdgenxo tool uses the capacitance solution from the original run.

Creating an ITF File

The following procedure provides an overview of ITF file creation. For more information about specific ITF statements, see Chapter 15, ITF Statements.

1. Specify the `TECHNOLOGY` statement:

   ```
   TECHNOLOGY = process_name
   ```

   The `TECHNOLOGY` statement is mandatory and should precede all other statements, but it does not need to be the first line. The *process_name* argument becomes the file name of the nxtgrd file.

2. Specify process description information. The keywords are listed in Table 52 and have names that suggest their intended uses. You must follow the rules for the values allowed.

   This information is required for all nxtgrd files that contain a `MULTIGATE` statement or a `WMIN` value less than or equal to 8 nm for any conductor layer.

Feedback

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the `TECHNOLOGY` statement
- Specify the keywords in the correct order

Table 52 Process Description Keywords

Keyword	Type	Description
`PROCESS_FOUNDRY`	String	Identifies the foundry
`PROCESS_NODE`	Float	Represents the process node
`PROCESS_TYPE`	String	Identifies the process type
`PROCESS_VERSION`	Float	Represents the process version
`PROCESS_CORNER`	String	Names the process corner

3. (Optional) Specify the global temperature and the relative permittivity of the background dielectric:

   `GLOBAL_TEMPERATURE = temp_value`

   `BACKGROUND_ER = relative_permittivity`

   The background permittivity fills the cross section with material of the given dielectric constant to an infinite height. Layer-specific permittivities specified in the ITF file override the global background dielectric. The default for the background dielectric is 1.0.

4. Define the basic layer characteristics for all of the conductor, dielectric, and via layers.

   You must specify the following layer characteristics:

   - Thickness of each conductor and dielectric layer
   - Minimum width and spacing of each conductor layer (design rule spacing)
   - Resistivity of each conductor layer
   - Permittivity of each dielectric layer
   - Resistivity information of each via layer
   - Connectivity information of each via layer

5. Specify additional ITF statements to model process effects.

Feedback

The following example shows a basic ITF file:

```
TECHNOLOGY = SIMPLE
DIELECTRIC TOP { THICKNESS = 3.600 ER = 3.9 }
CONDUCTOR  M2  {
    THICKNESS = 0.250 WMIN = 0.5
    SMIN = 0.5 RPSQ = 0.05 }
DIELECTRIC D3  { THICKNESS = 0.300 ER = 3.9 }
CONDUCTOR  M1  {
    THICKNESS = 0.212 WMIN = 0.5
    SMIN = 0.5 RPSQ = 0.05 }
DIELECTRIC D2  { THICKNESS = 0.200 ER = 4.2 }
CONDUCTOR  POLY{
    THICKNESS = 0.100 WMIN = 0.3
    SMIN = 0.3 RPSQ = 10.0}
DIELECTRIC D1  { THICKNESS = 0.300 ER = 3.9 }
VIA via1 { FROM=M1 TO=M2 RHO=0.263 }
VIA polyCont { FROM=POLY TO=M1 RHO=0.352 }
VIA subCont { FROM=SUBSTRATE TO=M1 RHO=0.500 }
```

ITF Files for Transistor-Level Flows

Transistor-level ITF files differ from gate-level ITF files because transistor-level extraction requires connectivity information down to the diffusion or substrate layers.

A transistor-level ITF file can describe a planar process, in which a single polysilicon (poly) layer is used for both the gate and field poly structures. Alternatively, the process might be nonplanar, using separate gate and field poly layers.

Via layers can be used only to connect two conducting layers. Therefore you must separate a poly contact from a substrate contact or a diffusion contact.

Covertical layers (such as gate and field poly layers) should be vertically overlapping. If the bottom of the field poly layer is higher than the top of the gate poly layer, the two poly layers are not connected to each other. In this case you must create a dummy via layer to connect the two poly layers.

ITF File Example

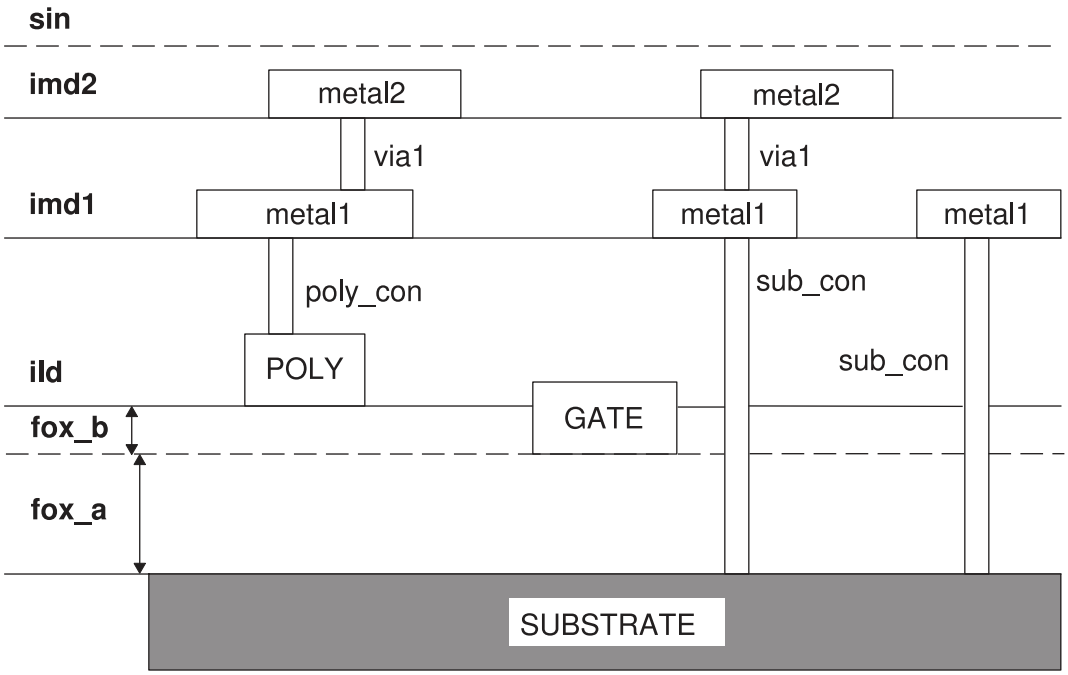
The following example shows the ITF statements that describe the cross section shown in Figure 124. For more examples, see Chapter 13, ITF Examples.

```
TECHNOLOGY=add4_2m1p

DIELECTRIC sin {THICKNESS=0.70 ER=7.9}
DIELECTRIC imd2 {THICKNESS=1.00 ER=4.2}
CONDUCTOR metal2 {THICKNESS=0.50 WMIN=0.35 SMIN=0.35 RPSQ=0.07}
DIELECTRIC imd1 {THICKNESS=1.05 ER=4.2}
CONDUCTOR metal1 {THICKNESS=0.35 WMIN=0.30 SMIN=0.30 RPSQ=0.08}
```

Feedback

```
DIELECTRIC ild    {THICKNESS=0.70 ER=4.1}
CONDUCTOR poly   {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_b {THICKNESS=0.10 ER=3.9}
CONDUCTOR gate {THICKNESS=0.20 WMIN=0.25 SMIN=0.25 RPSQ=5}
DIELECTRIC fox_a {THICKNESS=0.25 ER=3.9}

VIA via1     {FROM=metal1    TO=metal2 AREA=0.09 RPV=1}
VIA poly_con {FROM=poly      TO=metal1 AREA=0.04 RPV=12}
VIA sub_con {FROM=SUBSTRATE TO=metal1 AREA=0.04 RPV=16}
```

Figure 124 Process Cross Section



The Mapping File

A mapping file maps layers in the design database to layers in the nxtgrd file and is required for every StarRC run.

The following guidelines apply to mapping files:

- A database layer can map to only one process layer.
- Multiple database layers can map to the same process layer.

Feedback

- Database layers that map to the same process layer should not contain overlapping structures. Modify the LVS runset to use Boolean operations to avoid overlap and remove unused layers from the StarRC mapping file.
- Every logically connected database layer must be mapped in this file, even if the layer is derived or used only for intermediate connections with no real physical significance.
- In LEF/DEF flows, every layer defined in the technology LEF file—including vias—must be mapped.

Syntax conventions for mapping files are as follows:

- Commands are not case-sensitive. The convention in this user guide is to use lowercase for mapping file commands.
- Layer names are case-sensitive.
- Characters after an asterisk are considered to be part of a comment.

By default, the StarRC tool does not check the consistency of layers specified in a mapping file and layers specified in a GDSII, OASIS, or nxtgrd file. You can enable these checks by setting the `REPORT_UNMAPPED_GRD_LAYERS` or `REPORT_UNMAPPED_GDS_OASIS_LAYERS` commands to `YES`.

For more information about mapping files, see Chapter 16, Mapping Files. The following is an example of a mapping file:

```
conducting_layers    * Note the use of a single poly layer for all gates
    fpoly            Poly
    Bulk             SUBSTRATE
    Poly             Poly
    Rpoly            Poly
    Rndiff           Od
    Ndif             Od
    Nwell            SUBSTRATE
    ngate1           Poly
    Ngate            Poly
    Pgate            Poly
    Nsd              Od
    Psd              Od
    Pdif             Od
    Met4             Metal4
    Met3             Metal3
    Met2             Metal2
    met1             Metal1
via_layers
    Diffcnt          Odcont
    Plycnt           polyCont
    diffcnt          odCont
    m3via            via3
    m2via            via2
    mvia             via1
```

Feedback

```
marker_layers
    met1_pin
    met2_pin
    met3_pin
remove_layers
    cont
    diffcnt
ignore_cap_layers
    dngate ngate ngate1
    pdif psd
    ndif rndiff nsd
    Ngate           Nsd L=0.1
    Pgate           Psd L=0.1
    Nsd             SUBSTRATE L=0.1
    Psd             SUBSTRATE L=0.1
```

Feedback

The grdgenxo Command

Use the `grdgenxo` command at the operating system prompt to run the grdgenxo tool. The primary use of the grdgenxo tool is to create an nxtgrd file from an ITF process description file. You can also use the grdgenxo tool to perform a limited set of updating operations on existing nxtgrd files and to create TLUPlus files.

By default, nxtgrd files contain TLUPlus model information. The same nxtgrd file (also known as a common technology file) can be used both by the StarRC tool, which ignores commands that are specific to TLUPlus models, and by place-and-route or synthesis tools, which can use the TLUPlus model commands.

The following topics describe how to use the grdgenxo tool to achieve different goals:

- General Options
- Generating a Combined nxtgrd and TLUPlus File
- Generating a Standalone TLUPlus File
- Updating an Existing nxtgrd File
- Creating an ITF File From an nxtgrd File or Another ITF File
- Updating the Resistance Parameters of an nxtgrd File
- Updating the Half-Node Scale Factor of an nxtgrd File
- Using Distributed Processing With the grdgenxo Tool

General Options

The grdgenxo command offers several options that provide general information.

The syntax is as follows:

```
grdgenxo
   [-h]
   [-usage]
   [-parse itf_file]
   [-check_smc_compatibility itf_smc -corner_files list_of_itf_files]
```

Argument	Description
`-h`	Shows command-line options
`-usage`	Shows command-line options
`-parse itf_file`	Checks an ITF file for correct construction

Feedback

Argument	Description
`-check_smc_compatibility` *`itf_smc`*	Checks an ITF file for compatibility with simultaneous multicorner (SMC) extraction requirements
`-corner_files` *`list_of_itf_files`*	Checks one or more ITF files for SMC extraction compatibility with the reference file *itf_smc*

The `-check_smc_compatibility` and `-corner_files` options check whether ITF files are suitable for SMC extraction. These options must be used together.

- The `-check_smc_compatibility` option takes a single ITF file name as an argument and checks whether that file contains any ITF commands that are not supported for SMC extraction. The specified file is also the reference file for the `-corner_files` option.
- The `-corner_files` option takes a list of space-delimited ITF file names as an argument and checks whether the files are compatible with the reference file specified by the `-check_smc_compatibility` option.

During SMC extraction, the StarRC tool issues error messages and stops if any of the corner nxtgrd files violate SMC requirements. Using the grdgenxo tool to check SMC compatibility issues in ITF files before generating nxtgrd files can save time later. The tool writes compatibility issues into a summary file named smc_compatibility.report, which lists the errors found for each ITF file.

For more information about corners file constraints, see Simultaneous Multicorner Extraction.

Generating a Combined nxtgrd and TLUPlus File

By default, nxtgrd files generated by the grdgenxo tool also contain TLUPlus model information so that the same file can be used by the StarRC tool and by place-and-route tools. The StarRC tool uses the nxtgrd information, while place-and-route or synthesis tools use the TLUPlus information to model process effects at early design stages.

The combined file can be read by the following tool versions:

- StarRC version M-2017.06-SP3 and later
- Fusion Compiler version N-2017.09-SP2 and later
- IC Compiler II version N-2017.09-SP2 and later

The syntax is as follows:

```
grdgenxo itf_file
   [-format ffile]
```

Feedback

```
[-link_device_models]
[-sct sct_file]
[-Version]
[-3dIcSubckt subckt_file]
[-set_max_num maxnum]
```

Argument	Description
itf_file	Input ITF file
-f *ffile*	Uses the specified format file for the TLUPlus model
-link_device_models	Reuses device models when they are composed of different conductors with identical capacitive properties.
-sct *sct_file*	Uses an SCT file (capacitance table file) provided by the foundry instead of generating new tables
-V (or -Version)	Includes the version number in the output file
-3dIcSubckt *subckt_file*	Input file for TSV subckt models
-set_max_num *maxnum*	Maximum number of entries in an ER_VS_SI_SPACING table. TLUPlus models allow a maximum of 64 table entries by default. If the ITF file contains an ER_VS_SI_SPACING table with more than 64 entries, use this option to specify an integer greater than or equal to the number of table entries. This option is not valid for standalone TLUPlus files.

Feedback

Combined File Limitations

The following commands are allowed in nxtgrd files but are not used in TLUPlus files. If the ITF file contains these commands, the grdgenxo tool generates a common technology file that includes a TLUPlus model without the commands and an nxtgrd model with the commands.

- `DROP_FACTOR`
- `DROP_FACTOR_LATERAL_SPACING`
- `ETCH_VS_CONTACT_AND_GATE_SPACINGS`
- `DIELECTRIC_FILL_EMULATION_VS_SI_SPACING`

The following commands are allowed in TLUPlus files but are not used in nxtgrd files. If the ITF file contains these commands, the grdgenxo tool generates a common technology file that includes two TLUPlus models, one with the commands and one without the commands, and an nxtgrd model without the commands.

- `FILL_RATIO`
- `FILL_SPACING`
- `FILL_TYPE`
- `FILL_WIDTH`

Feedback

Generating a Standalone TLUPlus File

TLUPlus models are used by place-and-route or synthesis tools for modeling process effects at early design stages. The TLUPlus model file is in a binary format and contains an ASCII header section that lists the input files used.

By default, nxtgrd files generated by the grdgenxo tool contain TLUPlus information so that the same file can be used by the StarRC tool and by place-and-route tools. If necessary, you can also create a standalone TLUPlus file from an ITF file.

The syntax is as follows:

```
grdgenxo
   -itf2TLUPlus
   [-format ffile]
   -i itf_file
   -o tlu_file
```

Argument	Description
`-itf2TLUPlus`	Generates a TLUPlus model from an ITF file. If used, this argument must be specified first.
`-f ffile`	Uses the specified format file when creating TLUPlus models
`-i itf_file`	Input ITF file
`-o tlu_file`	Output TLUPlus file

The grdgenxo tool searches for the STAR-RC2-TCAD license to enable TLUPlus output. If a STAR-RC2-TCAD license is found, a Galaxy-Common license is not necessary. If a STAR-RC2-TCAD license is not found, the grdgenxo tool looks for a Galaxy-Common license. If neither license is found, an error message is issued.

A directory named *technology_name*.TLUPlus contains the temporary results. The input ITF file (extension .itf) and the format file (extension .format) are saved in that directory. If subsequent grdgenxo runs are performed on the same database, the grdgenxo tool checks the input files for changes from previous runs and issues an error message if there are any differences.

The following ITF keywords are not supported for TLUPlus models:

- `DROP_FACTOR`
- `DROP_FACTOR_LATERAL_SPACING`
- `ETCH_VS_CONTACT_AND_GATE_SPACINGS`

Feedback

The grdgenxo tool uses the following defaults for TLUPlus models:

- Units of fF, ohm, and microns (place-and-route tools convert units as necessary)
- Nominal operating conditions for capacitance tables names
- Capacitance tables dimension of 5x16 for width versus spacing
- Capacitance tables grid points that are multiples of minimum width and spacing values

To change the defaults, create a format file that contains the parameters shown in Table 53. You do not have to specify parameters for every layer. You do not need to specify the `WIDTH` and `SPACING` values if you want to change only the capacitance table dimensions while keeping the default width and spacing intervals.

Table 53 TLUPlus Format File Parameters

File content	Definition
`CAP_UNIT 1e-15`	Capacitance units (femtofarads). Do not change.
`RES_UNIT 1`	Resistance units (ohms). Do not change.
`OPCOND NOM`	Operating condition; can be `MIN`, `NOM`, or `MAX`
`LAYER poly`	Layer name from the ITF file (for example, poly, metal1, metal2) to which the subsequent statements apply.
`NUMWIDTH 3`	Number of capacitance table width points for the named layer; can be 2 to 16 inclusive.
`NUMSPACING 3`	Number of capacitance table spacing points for the named layer; can be 2 to 16 inclusive.
`WIDTH 0.15 0.3 0.45`	Values of the capacitance table width points.
`SPACING 0.15 0.3 0.45`	Values of the capacitance table spacing points.
`LAYER metal2`	Layer name to which the subsequent statements apply.
`NUMWIDTH 5`	Number of capacitance table width points for the named layer.
`NUMSPACING 16`	Number of capacitance table spacing points for the named layer.

Feedback

You must change the defaults specified in the TLUPlus files in the following cases:

- To create TLUPlus tables for minimum or maximum operating conditions
- To change the dimension of the capacitance table. Larger tables do not necessarily provide greater accuracy, but smaller tables reduce runtime.
- To use nonuniform width and spacing values, which might improve accuracy by reducing the need for interpolation or extrapolation

Dimensions of resistance tables and width points, if applicable, are determined automatically based on the information in the ITF file.

Updating an Existing nxtgrd File

The `-inc` option of the `grdgenxo` command allows you to update an nxtgrd file based on a modified ITF file. You must supply the modified ITF file in the working directory. The grdgenxo tool updates the nxtgrd file based on the differences between the two ITF files.

The syntax is as follows:

```
grdgenxo
   -inc
   [-old_itf old_itf_file]
   new_itf_file
```

Argument	Description
`-inc`	Performs an incremental update of the nxtgrd file. The grdgenxo tool recognizes the modified layers in the ITF file during successive runs and regenerates the capacitance models for the changed layers only.
`-old_itf` *`old_itf_file`*	Original ITF file. This option is not usually required.
`new_itf_file`	Modified ITF file

You can run the grdgenxo tool incrementally on a previous run directory only when using the same version of the grdgenxo tool, and only if the changes you have made to the ITF file do not affect the capacitance tables. Changes that modify conductor resistance values are allowed.

In a distributed processing environment, the grdgenxo tool performs additional checking to verify that all processors are using the same ITF file.

Feedback

Limitations

Some commands are primarily related to conductor layers and have only localized effects on capacitance values. Therefore, you can make changes to these commands and perform an incremental update to an existing nxtgrd file. Changes to the following commands in the `CONDUCTOR` definition are supported:

```
AIR_GAP_VS_SPACING
CAPACITIVE_ONLY_ETCH
ETCH
ETCH_VS_WIDTH_AND_SPACING
ETCH_VS_WIDTH_AND_SPACING: CAPACITIVE_ONLY
ETCH_VS_WIDTH_AND_SPACING: ETCH_FROM_TOP
MEASURED_FROM
SIDE_TANGENT
SMIN
THICKNESS
WMIN
```

Global layer options have extensive effects on capacitance values. Changes to the following commands in the `CONDUCTOR` definition are not supported:

```
DROP_FACTOR
FILL_RATIO
FILL_SPACING
FILL_WIDTH
FILL_TYPE
GATE_TO_CONTACT_SMIN
```

Any changes to dielectric layers have large effects on capacitance values. Do not use the incremental option for the following types of changes:

- Changes to the total number of conductors or dielectrics
- Removal of a conductor

Inappropriate `WMIN` and `SMIN` values specified in the ITF file might cause unwanted opens or shorts of the neighboring layers by applying the etch values. A message is issued during the grdgenxo run for `WMIN` violations. Reporting of `SMIN` violations is off by default; to enable it, use the `REPORT_SMIN_VIOLATION: YES` command.

Feedback

Creating an ITF File From an nxtgrd File or Another ITF File

An nxtgrd file contains the original ITF commands that were used to generate it. You can extract the ITF commands from the nxtgrd file. If you extract ITF information from an nxtgrd file that you receive from a foundry, you might find that part or all of the ITF information is encrypted, in which case it is presented as random ASCII characters.

If you later create an nxtgrd file from an ITF file with encrypted fields, the grdgenxo tool issues an error message if it detects that the originally encrypted fields have been modified.

The syntax for extracting the ITF information is as follows:

```
grdgenxo
   -nxtgrd2itf
   -i nxtgrd_file
   -o itf_file
```

Argument	Description
`-nxtgrd2itf`	Extracts the ITF commands from an existing nxtgrd file
`-i nxtgrd_file`	Input nxtgrd file
`-o itf_file`	Output ITF file

You can fully or partially encrypt an ITF file and store the results in a new ITF file. Edit the ITF file as described in Encrypting ITF Information, then use the following syntax to create a new (encrypted) ITF file from the old ITF file:

```
grdgenxo
   -itf2itf
   -i old_itf_file
   -o new_itf_file
```

Argument	Description
`-itf2itf`	Extracts the ITF commands from an existing ITF file
`-i old_itf_file`	Input ITF file
`-o new_itf_file`	Output ITF file

Feedback

Using the ITF Graphic Viewer

The ITF graphic viewer visual debug tool allows you to view the ITF stack and assess the correctness of an ITF file during the creation process and for further iterations.

The following command launches the ITF graphic viewer with the specified ITF file:

```
grdgenxo -viewer itf_file
```

Note:

To access help, use the **Help** menu command with a single key command. Click the question mark (?) to view a brief description of each key.

You can use the ITF graphic viewer to view the stacks of conductors, vias, and dielectrics from the specified ITF file. The ITF graphic viewer provides efficient visualization and query features to check the correctness of the ITF file and to debug issues. The basic flows of the ITF viewer includes the following functions:

- Parsing of ITF (similar to `-parse`) and exits on errors
- Identifying different possible stacks and generating input data for visualizing
- Launching `grdgenxo` in visualization mode with graphic viewer

Performing Operations With Mouse Controls

The following operations can be performed by clicking (left-click or right-click) and followed by dragging using the mouse before releasing, as shown in Table 54:

- Left-click at any point on the graphic viewer to move the layer stacks either up or down or either left or right.
- Right-click on the conductor or via layers to select the layers.
- Position the mouse for measuring distances.

Table 54 Mouse actions with descriptions

Keyboard Command	Task
Left-click and release	Moves the stack up, down, left, or right
Click and drag horizontally or vertically	Shows the dragged distance
Click and drag diagonally	Zooms the specified rectangular view
Right-click on a conductor or via layer	Selects the layer

Feedback

Accessing Menus With Keyboard Shortcuts

The topics explains how to access the following ITF graphic viewer menus with the keyboard shortcuts :

- View Menu
- Show Menu
- Color Menu
- Markup Menu
- Layers Menu

View Menu

The **View** menu commands control the view scale and position of the ITF viewer. Table 55 displays the keyboard shortcuts that you can use to perform tasks of **View** menu commands.

Table 55 Keyboard Shortcuts to use View Menu Command

Keyboard Shortcut	Task
o	Zooms Out
i	Zooms In
h	Home
~	Toggles background between white and black
Control-o	Shows overlapped in the cross section

Show Menu

The **Show** menu commands change the objects that are displayed. Table 56 displays the keyboard shortcuts that you can use to perform tasks of **Show** menu commands.

Table 56 Show Menu Keyboard Commands

Keyboard Shortcut	Task
d (small case)	Highlights the next dielectric region
s	Highlights the previous dielectric region (*)

Feedback

Table 56 Show Menu Keyboard Commands (Continued)

Keyboard Shortcut	Task
D (capital case)	Toggles to display dielectric regions (*) `Permittivity, Layer (*), Hidden`
t	Toggles to display trapezoid shape of layers
*	Toggles whether to show the highlighted layer

Note:

1. By default, dielectrics are displayed with a pattern reflecting their `permittivity`, and `d` or `s` highlights the dielectrics corresponding to the next or previous ER value, displayed in the header line.
2. Pressing D one time enables the individual dielectrics visualization mode. Each individual dielectric has black contour lines and `d` or `s` highlights the next dielectric layer, which name is displayed with the ER (dielectric permittivity) in the header line.
3. Pressing `D` once more hides the dielectrics and pressing once more reverts to the default.

Color Menu

The **Color** menu commands change the color of the selected objects. To set colors on selected objects, use the following keyboard shortcuts:

- `Control-r`: Red
- `Control-g`: Green
- `Control-b`: Blue
- `Control-y`: Yellow
- `Control-c`: Cyan
- `Control-m`: Magenta
- `Control-d`: Black
- `Control-p`: Pattern is set to the next in series
- `Control-k`: Resets all color settings to the default.

Feedback

Markup Menu

The **Markup** menu commands allow to mark the current 2-D (top or side) view. Table 57 displays the keyboard shortcuts that you can use to perform tasks of **Markup** menu commands..

Table 57 Markup Menu Keyboard Commands

Keyboard Shortcut	Task
Control-t	Inserts text at cursor
Control-T	Inserts default text name for the selected layer at cursor
Control-X	Removes all text
@	Marks as the first measurement point
#	Marks as the additional measurement point
!	Ends measurement
=	Prints on the terminal a list of hidden and visible objects at the cursor position

Layers Menu

The **Layers** menu commands allow to select a layer to be colored or patterned (see Color Menu). Table 58 displays the keyboard shortcuts that you can use to perform tasks of **Layer** menu commands.

Table 58 Markup Menu Keyboard Commands

Keyboard Command	Task
l	Highlights the next layer
k	Highlights the previous layer (*)
L	Removes the highlight from layers
Control-l	Selects the layer by the name provided
Control-C	Prints conductor layers data in the tabular format
Control-D	Prints dielectric layers data in the tabular format
Control-V	Prints via layers data in the tabular format
Note that you can use * (in the Show Menu) to toggle if you need to display only the selected layer.	

Feedback

Selecting a Layer by Name

The `Control-l` command allows you to select a layer by name. The text you type is shown on the top-left of the graphics window. You can use the following keyboard keys:

- Backspace or Delete: Deletes the last character or returns.
- Enter or Return: Selects the indicated layer.
- Esc: Exits without selecting the layer.

Feedback

Encrypting ITF Information

If you extract ITF information from an nxtgrd file that you receive from a foundry, you might find that part or all of the ITF information is encrypted. If you create an nxtgrd file from an ITF file with encrypted fields, the grdgenxo tool issues an error message if it detects that the originally encrypted fields have been modified.

The presence of encrypted fields in the ITF file suppresses technology information and layer data in the nxtgrd header and produces a simple layer list instead of a full ITF command list in the file header.

Partial Encryption by Field

In the ITF file, mark a field to encrypt by prefixing the field with the pound sign (#). Use the `-itf2itf` option with the `grdgenxo` command to created an updated ITF file with partial encryption.

In the following ITF file, the # character indicates that the `THICKNESS`, `WMIN`, `SMIN`, and `RPSQ` values should be encrypted.

```
CONDUCTOR M3 {
   THICKNESS= #0.81
   WMIN= #0.61
   SMIN= #0.51
   RPSQ= #0.661
   FILL_RATIO=0.5
   FILL_WIDTH=0.5
   FILL_SPACING=0.25
   }
```

The following example shows the appearance of the ITF statements in the output nxtgrd file. Some fields are encrypted and others are unencrypted.

```
CONDUCTOR M3 {
   THICKNESS= @46e54570d7fccee0c7
   WMIN= @74fdf83f98c5d8d288
   SMIN= @22d884f1d2b9fe84c2
   RPSQ= @507cd4ceede9ddf1ece7
   FILL_RATIO=0.5
   FILL_WIDTH=0.5
   FILL_SPACING=0.25
   }
```

Partial Encryption by Section

You can encrypt sections of an ITF file by using the `#beginHide` and `#endHide` marker statements. These statements must be the only text on that line in the file.

The following example shows an ITF file after editing.

```
CONDUCTOR m3 {
     THICKNESS = 0.81
```

Feedback

```
#beginHide
 WMIN = 0.6
 SMIN = 0.5
 RPSQ = 0.433
#endHide
 FILL_RATIO = 0.5
 FILL_WIDTH = 0.5
 FILL_SPACING = 0.25
}
```

The following example shows how the partially encrypted ITF file might look after extraction from an nxtgrd file:

```
CONDUCTOR m3 {
     THICKNESS = 0.81
#beginHide
@46e54570d7fccee0c774fdf83f90c5d8d28822d884fld2b0fe84c2
#endHide
 FILL_RATIO = 0.5
 FILL_WIDTH = 0.5
 FILL_SPACING = 0.25
}
```

Feedback

Updating the Resistance Parameters of an nxtgrd File

You can update the resistance parameters of an nxtgrd file by using the `-res_update` option with the `grdgenxo` command in conjunction with a new ITF file. The syntax is as follows:

```
grdgenxo
   -res_update new_itf_file
   -i old_nxtgrd_file
   -o new_nxtgrd_file
```

Argument	Description
`-res_update new_itf_file`	Allows updating of resistance parameters from commands such as the `RPSQ` and `RHO_VS_WIDTH_AND_SPACING` commands without processing through the field solver
`-i old_nxtgrd_file`	The nxtgrd file to be updated
`-o new_nxtgrd_file`	The updated nxtgrd file

This method can save substantial computing time. The version number and time stamp are not updated; the original values are kept.

The following limitations apply:

- You can update existing resistance parameters, but you cannot introduce new resistance parameters into an nxtgrd file.
- You cannot change any StarRC commands that affect capacitance.

Feedback

Updating the Half-Node Scale Factor of an nxtgrd File

You can update the half-node scale factor of an nxtgrd file by using the `-add_sf` option with the `grdgenxo` command. The syntax is as follows:

```
grdgenxo
   -add_sf factor
   -i old_nxtgrd_file
   -o new_nxtgrd_file
```

Argument	Description
`-add_sf factor`	Reprocesses an existing nxtgrd file with a new half-node scaling factor
`-i old_nxtgrd_file`	Input nxtgrd file to be rescaled
`-o new_nxtgrd_file`	The updated nxtgrd file

If you generated an nxtgrd file without setting a half-node scale factor, or you would like to change the scale factor for an nxtgrd file, run the grdgenxo tool to generate an updated nxtgrd file. The following example sets the scale factor to 0.9:

```
% grdgenxo -add_sf 0.9 -i noshrink.nxtgrd -o shrink.nxtgrd
```

If you generated an nxtgrd file with a `HALF_NODE_SCALE_FACTOR` value and you would like to run extraction without the shrink, remove the scaling factor from the nxtgrd file by setting the factor to 1, as follows:

```
% grdgenxo -add_sf 1 -i noshrink.nxtgrd -o shrink.nxtgrd
```

See Also

- HALF_NODE_SCALE_FACTOR

Feedback

Using Distributed Processing With the grdgenxo Tool

The runtime of the grdgenxo tool can be greatly reduced by running the tool in distributed processing mode. You can run multiple grdgenxo processes concurrently on multiple cores. Each grdgenxo process executes a subset of tasks to create an nxtgrd file from an ITF process description file.

Use one of the following methods to start the grdgenxo tool in distributed processing mode:

- Manual Submission
- Automatic Submission

See Also

- GRD_DP_STRING
- ENABLE_IPV6
- NUM_CORES

Manual Submission

You can start multiple grdgenxo processes on the command line or by using a shell script. Run the grdgenxo tool on a single host or on multiple hosts by logging in manually, or by submitting jobs to a compute farm. In both cases, you must monitor the state of your jobs. Make sure to run the `grdgenxo` command from the same directory for all hosts. You can start additional grdgenxo processes any time by assigning them to the existing runs after launching the first process.

Automatic Submission

You can specify the number of jobs to run multiple grdgenxo processes concurrently by setting the `NUM_CORES` and `GRD_DP_STRING` commands in the DP configuration file. After you create the DP configuration file, start the grdgenxo tool in distributed processing mode by using the following command:

```
grdgenxo -dp_config dp_config_file itf_file <grdgenxo options>
```

When you create the DP configuration file, specify the following parameters to launch a supervisor process:

- Number of cores to use for distributed processing in the grdgenxo tool
- Method to get compute resources and submit remote worker process

Feedback

You must specify the arguments to the `grdgenxo` command in the following order to start the grdgenxo tool in distributed processing mode:

1. The first option must be `-dp_config`.
2. The path to the DP configuration file.
3. The path to the ITF file.
4. Other options can be specified as needed.

Note:

If you do not specify the options and arguments in the correct order, the grdgenxo tool issues an error message.

When you use the `grdgenxo -dp_config` command,

- You launch a single grdgenxo process to start the grdgenxo in distributed processing mode called the supervisor process.
- The supervisor process in turn starts remote jobs called the worker processes based on the settings specified in the DP confg file.
- The supervisor process monitors the state of all worker processes and terminates after the worker processes successfully create the nxtgrd file or if the worker processes fail.

  The supervisor process also terminates any pending jobs after the run is complete. Press **ctrl+c** to terminate any ongoing run to interrupt the supervisor process and the associated worker processes.

  You can execute only one supervisor process in a directory. If you execute more than one supervisor process from the directory, the grdgenxo tool issues an error message.

  All activities and messages in an automatic submission are saved in a log file with the detailed time-stamped logs, in the dp_logs directory within the common technology directory generated by the grdgenxo tool. It also includes the CPU load, memory usage, and disk space information
- The number of worker processes to be executed must be specified in the DP configuration file. Additional worker processes cannot be assigned to the existing runs after initiating the grdgenxo run.

Examples

The following examples show how to specify commands in the DP configuration file. For information to configure `ssh` protocol to use Example 17 and Example 18, see Supported Computing Platforms.

Example 17 Running 4 Jobs on a Local Machine

```
NUM_CORES: 4
GRD_DP_STRING: list localhost:4
```

Example 18 Running 2 Jobs on hostA and 2 jobs on hostB by Using ssh to Logon to These Hosts

```
NUM_CORES: 4
GRD_DP_STRING: list ssh hostA:2 hostB:2
```

Example 19 Running 32 jobs on the Compute farm using qsub

```
NUM_CORES: 32
GRD_DP_STRING: qsub -V -l -cwd -l mem_free=8g -P normal
```

12

Process Characterization

You can model specific process effects using StarRC commands and ITF statements.

For more information, see the following topics:

- FinFET Modeling
- RVTV Flow and Gate-All-Around FinFETs (GAAFETs) Extraction
- Through-Silicon Vias
- Double or Multiple Patterning Technology
- Conductor Layer Thickness Variation
- Bottom Conductor Thickness Variation
- Conductor Sheet Zones
- Tall Contact Modeling
- Tall Via Modeling
- Bridge Via Modeling
- Gate-To-Diffusion Capacitance Extraction
- Gate Conductor Resistance
- Conformal Dielectrics
- Conductor Cutting Through Dielectric
- Covertical Conductors
- Conductor Drop Factor
- Dual Polysilicon Gate Process
- Double-Polysilicon Process
- Layer Etch
- Overlapping Wells

- Damage Modeling
- Temperature Derating
- Half-Node Scaling
- Via Merging
- Diffusion Resistance
- User-Defined Diffusion Resistance

Feedback

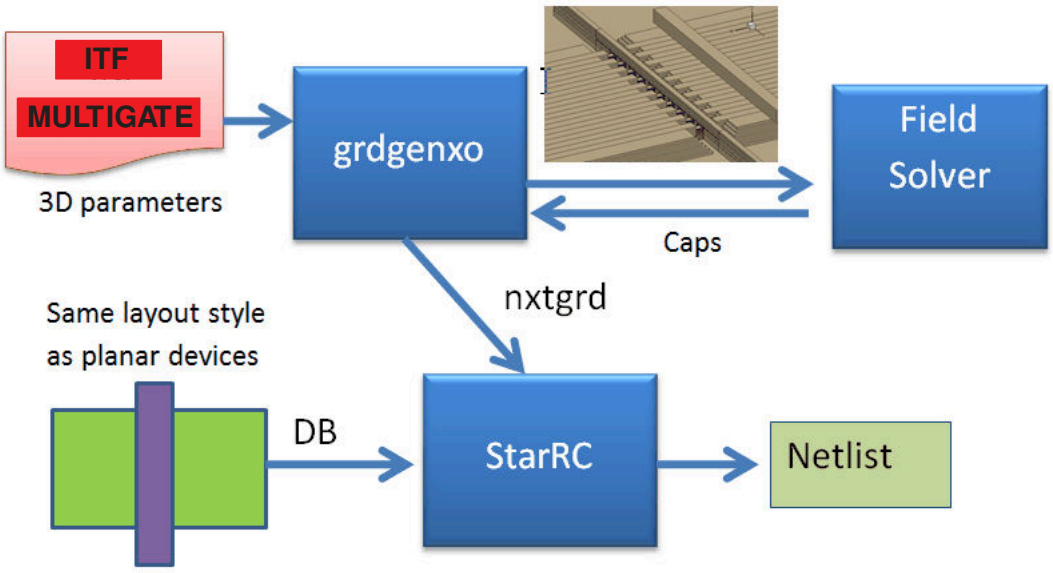
FinFET Modeling

StarRC extraction supports FinFET devices. FinFET device layout is similar to the planar device layout; in both cases the device layout includes the intersection of a diffusion layer with polysilicon.

For FinFET modeling, the ITF file contains a `MULTIGATE` block which describes how to map the device layout to 3-D geometries. The grdgenxo tool creates models of FinFET devices based on the `MULTIGATE` statement in the ITF file. These models are characterized using a 3-D field solver, and parameterized capacitance tables are built and stored in the nxtgrd file. The StarRC tool uses the capacitance tables during both standard extraction and field-solver extraction. Figure 125 shows the FinFET extraction flow.

To enable the FinFET extraction flow, add `MULTIGATE_MODELS: YES` to the StarRC command file.

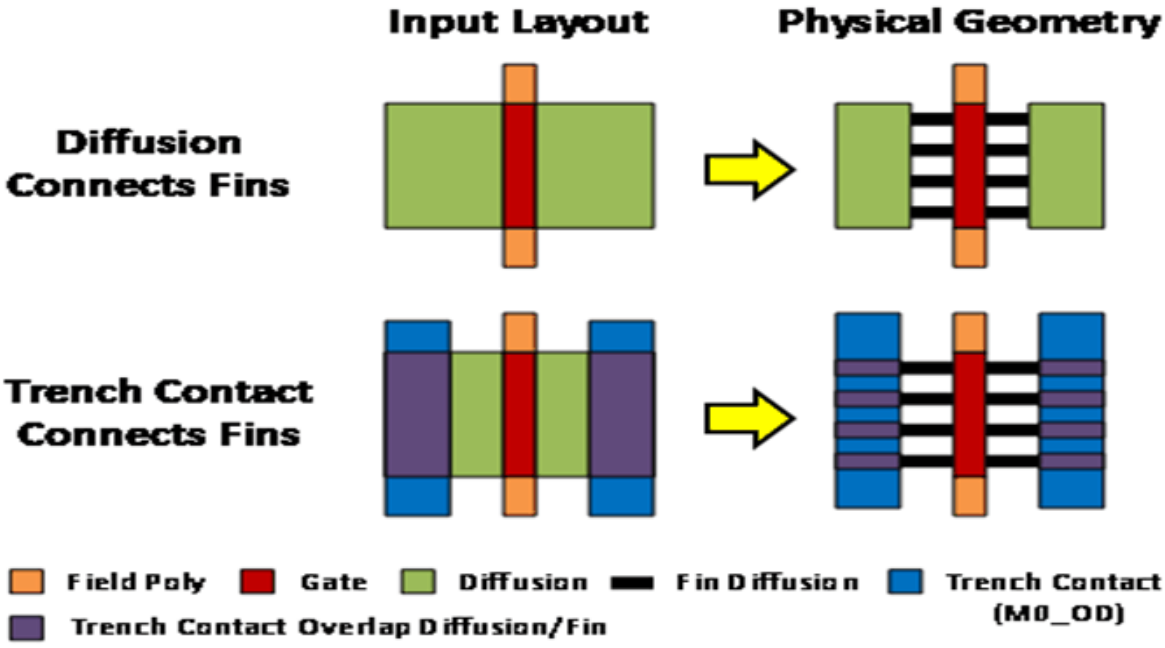
Figure 125 FinFET Extraction Flow



The StarRC tool recognizes FinFET devices and converts them to realistic FinFET physical geometries, as shown in Figure 126.

Feedback

Figure 126 FinFET Layout to Physical Geometry



FinFET ITF Statement Guidelines

Design Guidelines

- Set `RAISED_DIFFUSION_GROWTH` to a value greater than zero.
- When using `RAISED_DIFFUSION_GROWTH`, set the raised diffusion region in the diffusion layer.
- Set `RAISED_DIFFUSION_GROWTH` to less than one half the `FIN_SPACING` value.
- Use trench contacts with FinFETs. Via-style diffusion contacts are not supported.
- Represent FinFET poly and diffusion as noncovertical. To understand this requirement, see Process Description Requirements.

Some FinFET processes have highly nonlinear gate resistances due to the complex 3-D geometry of the gate conductors. To model this nonlinearity, provide the resistance per square (RPSQ) values in the `RPSQ_VS_SI_WIDTH_AND_LENGTH` table.

Process Description Requirements

FinFET processes intrinsically require covertical poly and diffusion layers because the gate extends down the sides of the transistor channel. However, the StarRC tool assumes that poly and diffusion layers are not covertical. Therefore, you must describe the FinFET

Feedback

poly and diffusion as non-covertical. Use the `RAISED_DIFFUSION_THICKNESS` statement to specify the full height of the diffusion conductor.

To model this process, define a very thin diffusion layer and a thick raised diffusion layer, as shown in Figure 127. Define a thin dielectric layer under the gate poly layer whose top surface is slightly higher than the top surface of the thin diffusion layer. Set the thickness of the gate poly layer to its actual thickness minus the thin dielectric under it. With this model, it is not possible to separately model etch effects on the raised diffusion layer, as shown in Figure 128. As a compromise, model the raised diffusion etch with an intermediate etch value applied to the full height of the diffusion conductor. If you are using a conformal dielectric layer, specify it to be in contact with its associated conductor. If the diffusion is made much thinner, any associated conformal dielectrics must begin at a correspondingly lower level, which might affect some processes.

Figure 127 FinFET Cross Section

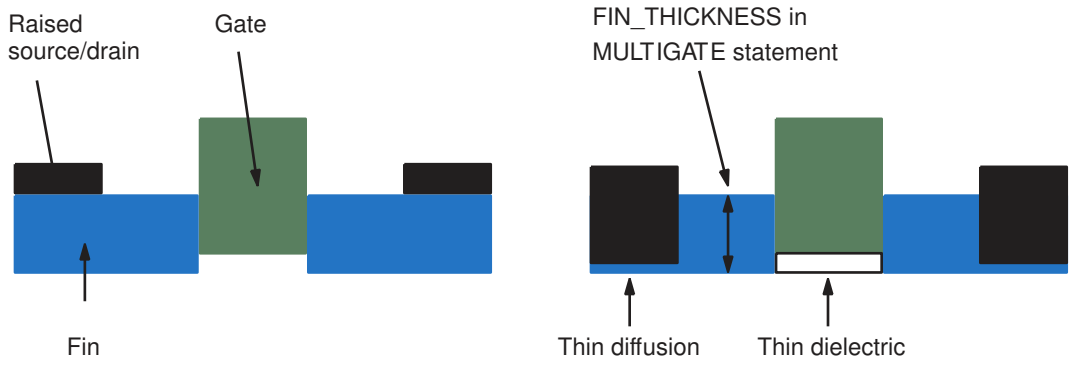
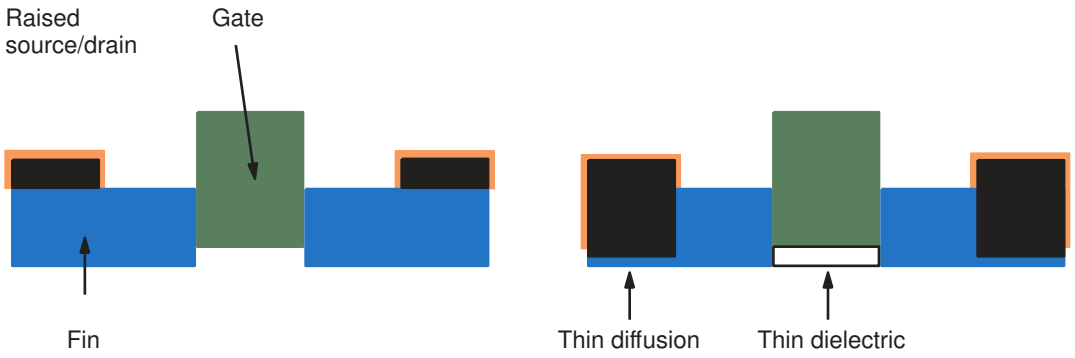


Figure 128 FinFET With Etch On Raised Diffusion Areas



Feedback

FinFET Capacitances

Figure 129 and Figure 130 show the capacitance components of a FinFET device.

Figure 129 FinFET Capacitance Components, Side View

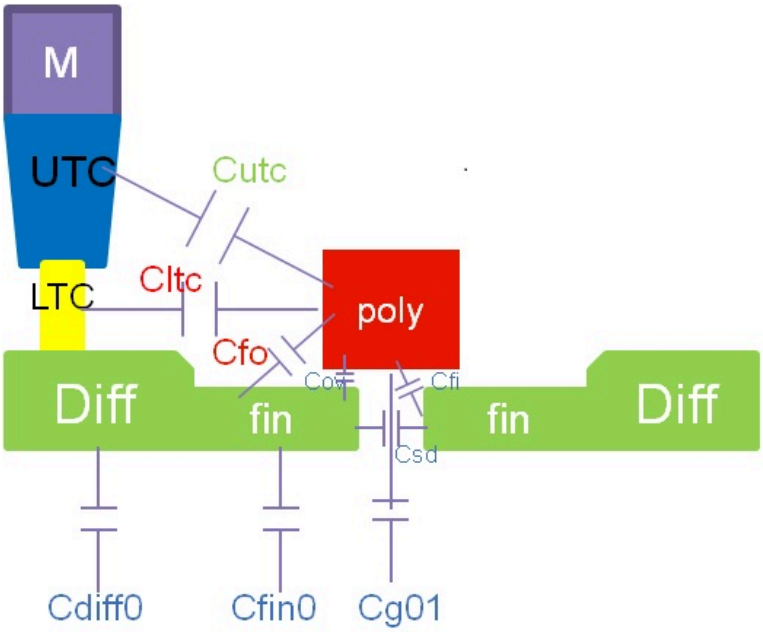
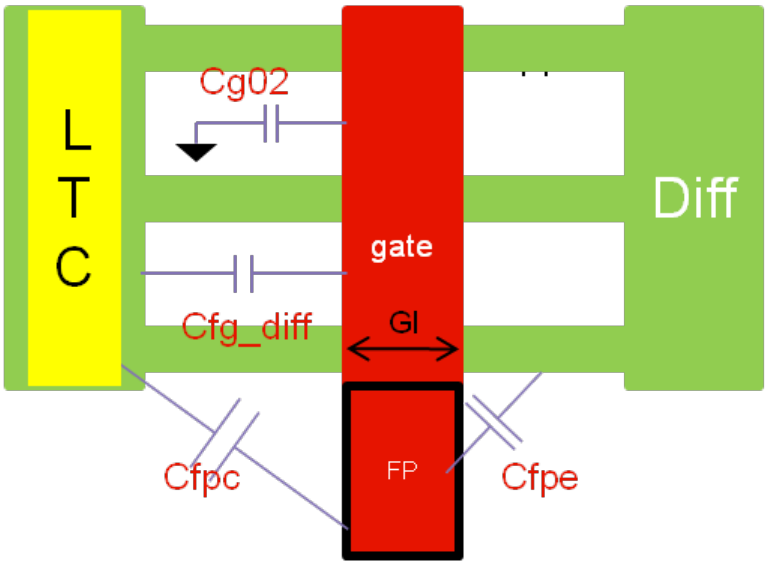


Figure 130 FinFET Capacitance Components, Top View



Feedback

The grdgenxo tool uses an internal 3-D field solver to build parameterized tables for each of the capacitance components. The tool uses the ITF `MULTIGATE` statement to construct the fin geometry, and assumes the `FIN_LENGTH`, `FIN_WIDTH`, and `FIN_SPACING` parameters are constant for each device type.

Four components are characterized for the capacitance tables. Labels in parentheses refer to Figure 129 and Figure 130:

- Cgd (gate to diffusion)

  This includes the capacitance from the gate to the top of the diffusion (Cfo) and from the gate to the fin and diffusion sidewalls (Cfg_diff). It does not include the overlap capacitance between the gate and the diffusion (Cov) or the capacitance from the gate to the diffusion through the channel (Cfi).

- Cg0 (gate to substrate)

  This includes the gate to substrate capacitance (Cg02) but not the gate to substrate capacitance through the channel (Cg01).

- Cgc (gate to contact)

  This is the same as the gate to lower trench contact capacitance Cltc in Figure 129. The upper trench contact capacitance Cutc is extracted outside of the grdgenxo tool.

- Cfpe (field poly to diffusion)

  This is labeled as Cfpe in Figure 130.

You can provide a table to represent gate-to-diffusion capacitance inside the channel (Cfi) by using the `GATE_TO_DIFFUSION_CHANNEL_CAP` statement in the ITF file.

If the StarRC command file includes the `IGNORE_GATE_CHANNEL_CAPACITANCE: NO` command, the total gate-to-diffusion capacitance (Cf) is extracted by the field solver and no Cfi capacitance subtraction occurs, regardless of whether a Cfi table exists.

Alternatively, you can calculate Cfo by first having the field solver extract the total capacitance Cf, then subtracting the Cfi value (interpolated from the Cfi table) from Cf. To enable this calculation method, use the `IGNORE_GATE_CHANNEL_CAPACITANCE: YES` command in the StarRC command file. If a Cfi table does not exist, the command has no effect.

RVTV Flow and Gate-All-Around FinFETs (GAAFETs) Extraction

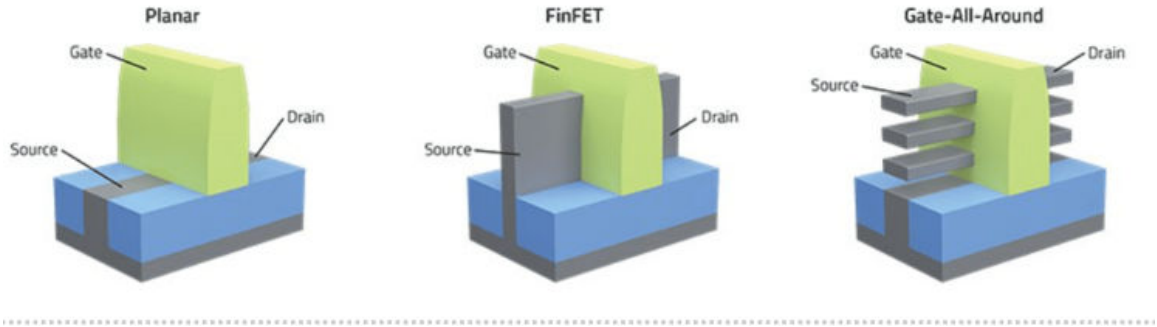
The advanced nodes technologies replaced FinFET devices with Gate-All-Around FinFETs (GAAFETs) nanowire or nanoribbon devices as shown in Figure 131.

In GAAFETs, a stack of lateral nanowires wrapped by gates from all the side is used to improve on-current (I_{on}) or off-current (I_{off}) performance and short channel margin.

Feedback

Additionally, the use of lateral nanowires or nanosheets has the advantage of a process flow that is not disruptive compared to FinFET processing.

Figure 131 Planar Verses FinFET Verses GAA technologies



You have the following advantages:

- The use of lateral nanowires or nanosheets has the advantage of a process flow that is not disruptive compared to FinFET processing.
- You can place devices in the middle of the interconnect stack
  - Without any substrate and device bulk layers (three-terminal devices).
  - With M0 and above layers - Used for signal routing (frontside).
  - BM0 and below layers - Used for routing a power grid.

Thick conductors are used to connect the frontside and backside interconnect stacks. Having lower-resistance BM* layers for routing the power signals allows designers to achieve better performance, lower cost, and improved chip endurance.

However, additional parasitic might be introduced because the conductors connecting backside to frontside are very close to the device region and have large thickness and patterning. Similarly, 3DIC approaches with through-slicon via (TSV) do not offer the required level of accuracy.

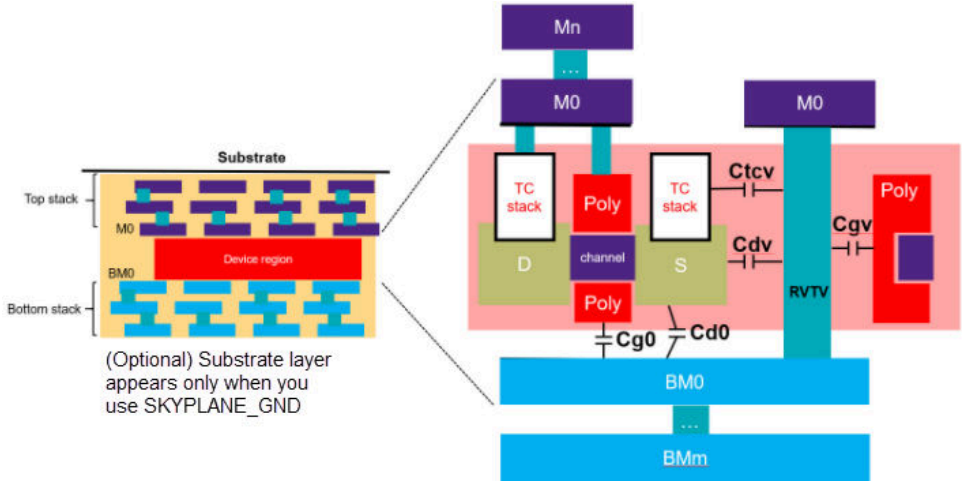
The StarRC tool provides the RVTV flow to accurately and efficiently describe this kind of process. In Supported Types of RVTV Connections, RVTV denotes the special conductor connecting backside layers to frontside layers.

Figure 132 shows a connection from the RVTV metal layer to the frontside metal layer (M0). The tool also allows connection to trench contact layers (extensions outside diffusion layers) and diffusion layers (backside trench contact).

RVTV-GAAFET flow is designed for the StarRC-QuickCap tool. The QuickCap tool is used for 3-D device models in nxtgrd file and for golden reference. ITF and QTF files must be setup correctly to allow a correct establishment between the tools.

Feedback

Figure 132 Interconnection of the Sack Around the Device Region



The main differences compared to a 3DIC TSV ITF are:

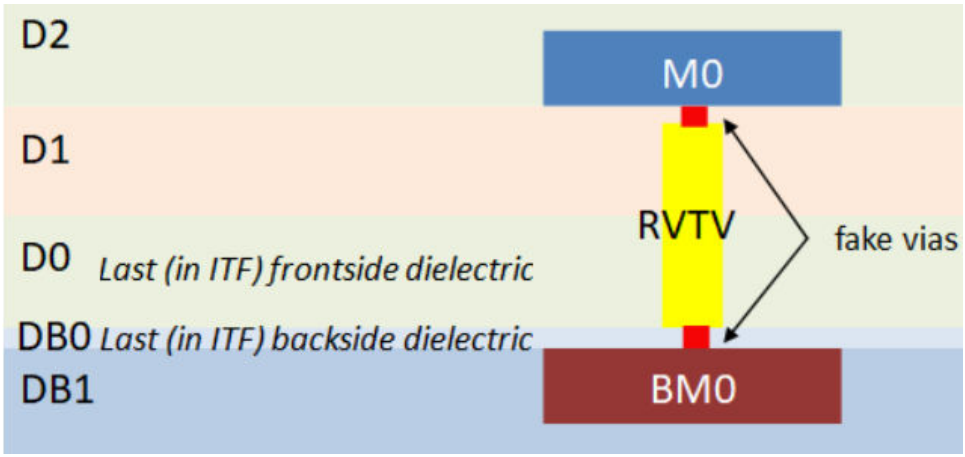
- (Optional) Specify the `SKYPLANE_GND` keyword to indicate the presence of the substrate layer at the top of frontside metal layer:
  - Specify `SKYPLANE_GND` after specifying the `TECHNOLOGY` keyword, the foundry process tags, and other global statements like `BACKGROUND_ER`, `GLOBAL_TEMPERATURE`, or `REFERENCE_DIRECTION`.
  - Specify`SKYPLANE_GND` before specifying the first dielectric or conductor layers.

  If you do not specify `SKYPLANE_GND` the ideal ground is considered as far away on the backside, and the tool isolates automatically from the regular dielectric stack using a thick air dielectric.
- Specify the RVTV statement to define the thickness of conductor that connects the backside and frontside metal layers.
  - Use the same arguments of the `CONDUCTOR` statement with the `RVTV` statement.
  - Add fake vias to connect RVTV conductor to the frontside metal layer (M0) and backside metal layer (BM0).
  - Position RVTV in the interconnect stack, because the last conductor of the frontside metal layers with respect to the bottom height in the interconnect stack implies matching of thicknesses for the planar dielectrics. Figure 133 indicates a similar scenario. Where,
    - D0 is the last dielectric in the fronside metal layer of an ITF statement.
    - RVTV is defined after the frontside metal layer.

Feedback

- DB1 appears before BM0 in the backside metal layer.
- DB0 dielectric is defined at the end to match with the thickness of the fake vias connecting the RVTV and BM0 layers.

Figure 133 Positioning of RVTV in Interconnect Stack



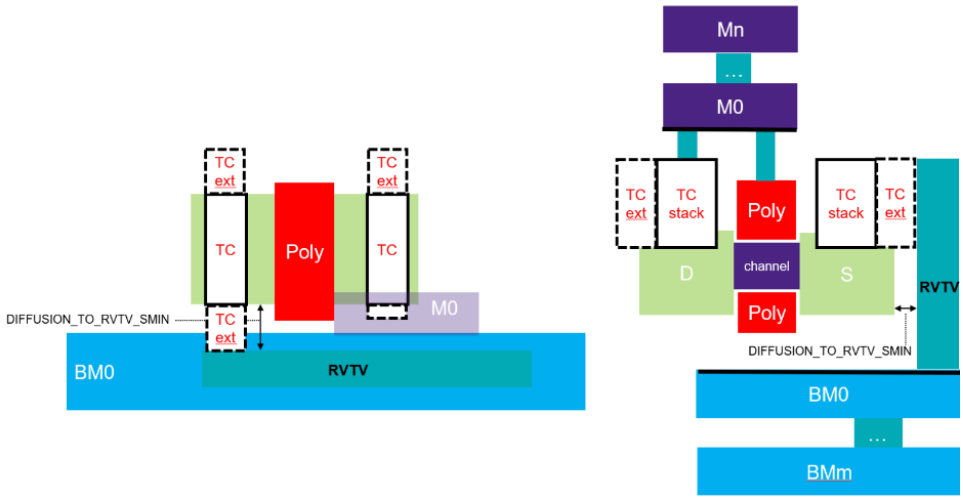
Supported Types of RVTV Connections

RVTV conductors typically connects the first backside layer close to the device region (BM0) to the front side layer. Supports the following types of connections:

- Connection to M0 (Figure 133).
- Connection to trench contact, either through fake via or by abutment and connection in LVS.

  The trench contact layer corresponds to the trench contact extension, outside the diffusion region as shown in Figure 134. This can apply to one-step or two-steps trench contact process. (For two-steps, the connection is to the top trench contact extension.

Figure 134 RVTV Connection to Trench Contact Layer



- Connection to the diffusion bottom with a fake via. In this case, RVTV is the backside trench contact as shown in Figure 135.

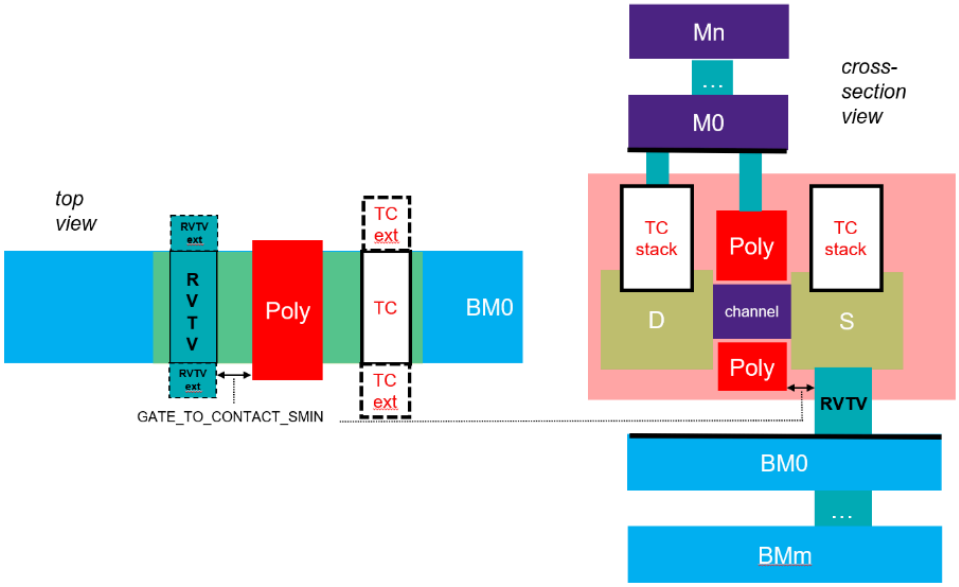
  You can define several RVTV conductors in one ITF file, either same type of connection for different `DEVICE_TYPE` settings, or different type of connections.

  If connection to the diffusion bottom with a fake via, similar to trench contact conductor, you can define RVTV extensions associated to the RVTV backside trench contact.

Feedback

If you can specify in the RVTV extension layer the associated RVTV conductor with `ASSOCIATED_TO` keyword.

Figure 135 RVTV Connection to Diffusion Layer



Examples

The following example illustrates a RVTV process. The frontside metal layers are defined before the RVTV statement and the backside metal layers are defined after the RVTV statement. The frontside metal layer farthest from the device region region (BM5) is defined first and the backside metal layer closest to the device region (BM0) is defined last.

```
TECHNOLOGY = rvtv_process
GLOBAL_TEMPERATURE = 25.0
BACKGROUND_ER = 4.1
REFERENCE_DIRECTION = HORIZONTAL

SKYPLANE_GND

$ Frontside ITF statements
CONDUCTOR M1 {
                THICKNESS=0.3 WMIN=0.12 SMIN=0.12 SIDE_TANGENT=0.2
                CRT1=7.0e-03 CRT2=-8.0e-07
             }
CONDUCTOR poly {
                THICKNESS=1.00 WMIN=0.13 SMIN=0.15
                RPSQ=0.015 GATE_TO_CONTACT_SMIN=0.08
               }
CONDUCTOR DIFFUSION {
                      THICKNESS=0.1 WMIN=0.1 SMIN=0.06
```

Feedback

```
                      CRT1=8.0e-03 CRT2=-1.0e-07 RPSQ=36.0
                    }

DIELECTRIC ILD_B        { ER=5.5 THICKNESS=0.5 }
DIELECTRIC GATE_OXIDE   { ER=4.3 THICKNESS=0.03 }
DIELECTRIC FOXIDE_A     { ER=5.0 THICKNESS=0.1 }

VIA via1                { FROM=M1 TO=M2 AREA=0.03 RPV=2.5 CRT1=3.0e-04
 CRT2=-6.0e-06 }
MULTIGATE SECTIONS fin {
                        }
$ RVTV statement
  RVTV rvtv1 {
        THICKNESS=0.0662 WMIN=4.0 SMIN=4.0
        CRT1=1.0e-03 CRT2=-7.0e-07
        DIFFUSION_TO_RVTV_SMIN=0.013
        }

CONDUCTOR BM5 {
                  THICKNESS=0.1 WMIN=0.1 SMIN=0.06
                  CRT1=8.0e-03 CRT2=-1.0e-07 RPSQ=36.0
                }

$ Backside ITF statements
CONDUCTOR BM0 {
                  THICKNESS=0.1 WMIN=0.1 SMIN=0.06
                  CRT1=8.0e-03 CRT2=-1.0e-07 RPSQ=36.0
                        }
VIA via1 { FROM=M1 TO=M2 AREA=0.03 RPV=2.5 CRT1=3.0e-04 CRT2=-6.0e-06 }
```

See Also

- SW_T
- MULTIGATE
- RVTV

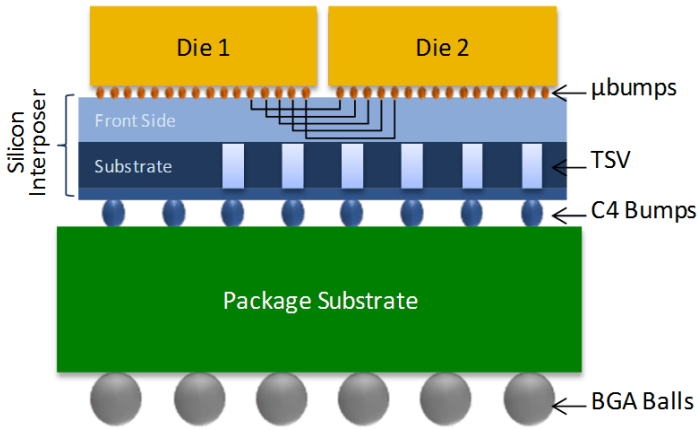
Feedback

Through-Silicon Vias

A through-silicon via (TSV) is a via that extends completely through a silicon substrate. One application is wafer-level integration, a packaging technique in which integrated circuits are connected to each other and to a package through a thinned silicon substrate called an interposer.

Figure 136 shows a silicon interposer attached to two die and a package. Conductive microbumps on the die connect to metal wires on the front side of the interposer. Through-silicon vias connect metal lines on the interposer front side to metal lines on the interposer backside, which are in turn connected to the package through metal bumps (labeled as C4 bumps in the figure).

Figure 136 Interposer Structure Showing TSVs and Microbumps



The StarRC tool can extract TSVs or microbumps that are defined as either cells or vias in the GDSII, LEF/DEF, Milkyway, and IC Compiler II (NDM format) flows. A process can include more than one TSV definition. For example, one type of TSV might connect frontside layer M1 and backside layer BM1, while another type of TSV connects frontside layer M2 to backside layer BM2.

TSV extraction requires the following general steps:

1. Use the `TSV` statement in the ITF file to define each type of through-silicon via. Separate `TSV` statements must be used for each type of TSV.
2. Use the `3D_IC_SUBCKT_FILE` command in the StarRC command file to specify parasitics models for each type of TSV. This command can specify more than one file; each file can contain more than one model. If a model is specified multiple times, the StarRC tool uses the first model read and issues warnings for subsequent instances of the same model name.

3. Set the `3D_IC` command in the StarRC command file to `YES`.
4. Specify optional extraction conditions with the `3D_IC_FILTER_DEVICE`, `3D_ID_FLOATING_SUBSTRATE`, `3D_IC_TSV_COUPLING_EXTRACTION`, and `TSV_CELLS` commands in the StarRC command file.

The following usage notes apply:

- The StarRC tool calculates coupling capacitance between through-silicon vias (TSVs) of the same type.
- The tool does not calculate coupling capacitance between TSVs of different types under the assumption that they are far apart and the coupling capacitance is negligible.

See Also

- 3D_IC
- TSV_ CELLS
- OPERATING_FREQUENCY
- TSV

Microbump Modeling

The microbump is a type of via layer. The microbump connects to the top metal layer and a thin (1 nm) pseudo-metal layer at the top of the stack. The pseudo-metal layer is defined with `LAYER_TYPE=BUMP`.

Define a microbump in the ITF file as shown in the following example:

```
VIA <ubump_name> {
     FROM=<RDL Metal Layer>
     TO=<Pseudo Metal layer>
     RPV=<R>
}
CONDUCTOR <name> {
     THICKNESS= 0.001
     WMIN=<value> SMIN=<value>
     RPSQ=0.00000001
     LAYER_TYPE=BUMP
}
```

Feedback

3D-IC Flow With Cell-Defined TSVs and Microbumps

You can extract TSVs and microbumps defined as cells in the Fusion Compiler or IC Compiler II (NDM format), LEF/DEF, Milkyway, and GDSII flows.

GDSII Flow

You can extract TSVs and microbumps as devices in the GDSII flow if you define them as cells in the runset. When you set `SKIP_PCELLS` on TSV and microbump devices, the StarRC tool does not extract anything inside these devices. The parasitics inside these devices are part of the device characteristics. The StarRC tool extracts only to the pins of the TSV or microbump cells. These pins appear as instance ports in the netlist file and the corresponding cell instances are listed in the Instance Section.

Fusion Compiler, IC Compiler II, LEF/DEF, and MilkyWay Design Flows

The TSV cell flow is similar to the `SKIP_CELLS` flow. For LEF/DEF and Milkyway designs, the TSV cell source is defined by the database in conjunction with the `TSV_CELLS` command. For Fusion Compiler or IC Compiler II (NDM format) designs, the design database identifies the TSV cells and the `TSV_CELLS` command is not used.

The StarRC tool supports net-to-port TSV cells from the IC Compiler or IC Compiler II or Fusion Compiler tool. The tool supports merging the same net in the front and back. To keep the one net to two ports output in the netlist, use the `SHORT_PINS:MIXED` command.

If you do not specify the `3D_IC_SUBCKT_FILE` command for a TSV cell, that TSV cell is translated as a skip cell and listed in the instance section. The StarRC tool determines the parasitics from the nxtgrd file. If you specify the `3D_IC_SUBCKT_FILE` command for a TSV cell, StarRC extraction replaces the TSV cells with the specified subcircuit file and outputs a single netlist. The microbump cells are processed the same as the TSV cells.

Feedback

3D-IC Flow With Via-Defined TSVs and Microbumps

You can extract TSVs and microbumps defined as vias in the Fusion Compiler or IC Compiler II (NDM format), LEF/DEF, Milkyway, and GDSII flows.

GDSII Flow

Because the interposer does not contain devices, nets have different top-level pins attached to them, which causes short errors in the LVS report. To fix the shorts, add a pseudo-resistor device at the ports to separate them from the net in the interposer. To remove the pseudo-resistor device, specify the device in the `3D_IC_FILTER_DEVICE` command, which flattens it in the netlist.

If you do not specify the `3D_IC_SUBCKT_FILE` for TSV vias, the StarRC tool extracts a two-port model and outputs a single netlist. The parasitics are determined by the factor in the nxtgrd file.

If you specify the `3D_IC_SUBCKT_FILE` for TSV vias, the tool replaces the TSV vias with the subcircuit file and outputs a single netlist.

Follow the same process for microbump vias.

Fusion Compiler, IC Compiler II, LEF/DEF, and MilkyWay Design Flows

The TSV and microbump vias can be defined in the design database. StarRC extraction merges the nets in the front and back. To keep the same net and the two ports in the netlist output, set the `SHORT_PINS` command to `MIXED`.

If the `3D_IC_SUBCKT_FILE` is not provided for a TSV via, the tool extracts the two-port model and outputs a single netlist. The parasitics are determined by the factor in the nxtgrd file.

If the `3D_IC_SUBCKT_FILE` is provided for a TSV via, the tool replaces the TSV vias with the subcircuit file and outputs a single netlist.

A microbump via can be replaced by using the `3D_IC_SUBCKT_FILE` command.

Feedback

Double or Multiple Patterning Technology

Advanced processes sometimes require a double or multiple patterning lithography process, which uses two or more masks to print closely spaced patterns. Layers that do not meet the minimum spacing requirements are split into separate masks, referred to as colors. The exposures from the masks are overlaid to print the single layer. The StarRC tool can extract parasitics created by multiple patterning. The terms double and multiple patterning are used interchangeably in this user guide.

Double-patterning extraction provides the following features:

- Exactly simulates the mask color misalignment and reports the final parasitic results
- Supports simultaneous multicorner technology
- Supports both gate-level and transistor-level designs
- Creates the nxtgrd database using the ITF process technology file along with statistically determined shift-impact measurements to compensate for misalignment shifts
- Reads color information for specified nets and reports parasitics for these nets, to allow scenario evaluation before colorizing the entire design (precolor flow)

The following sections describe how to extract parasitics from designs using double-patterning technology:

- Extraction for Double-Patterning Processes
- Estimating Parasitics for Critical Nets Using the Precolor Flow

Extraction for Double-Patterning Processes

You model double-patterning misalignment effects by using dielectric constant changes, which are controlled by the following two commands:

- `ER_VS_SI_SPACING`

  Add the `ER_VS_SI_SPACING` layer definition statement to the ITF process technology file.

- `DPT`

  Add `DPT: YES` to the StarRC command file.

For layout-versus-schematic (LVS) tools, color information resides as layers in the design database. For example, you define a layer as M1_color1 and M1_color2 in a Hercules or IC Validator XTR view. The LVS deck can include these color layers in the output. The interface from LVS tools enables the StarRC tool to recognize the color layers.

Feedback

For Milkyway tools, use the following commands to map color layers:

- `DPT_COLOR_GDS_FILE`: *`file_name.GDS`*

  This command specifies the name of the GDSII file containing the color layer polygons.

- `DPT_COLOR_GDS_LAYER_MAP_FILE`: *`file_name`*

  This command specifies the name of the file that maps the color layers in the design database to the GDSII layers.

To map the color layers in the design physical database to an nxtgrd file modeled layer:

1. Create a section named `color_layers` in the mapping file. For more information, see the color_layers description in Chapter 16, Mapping Files.
2. Add each database layer name in the `DPT_COLOR_GDS_LAYER_MAP_FILE` to the `color_layers` section in the mapping file.

Estimating Parasitics for Critical Nets Using the Precolor Flow

Use the precolor flow to determine parasitics for different layout options. Upstream tools create the `DPT_COLOR_GDS_FILE` using a precolor attribute in the design database.

The precolor flow is built around the simultaneous multicorner analysis feature. To enable a precolor flow with multiple corners,

1. Define a typical nxtgrd corner in the corners file using the NOMINAL keyword.
2. Use the `NOMINAL` corner when extracting the same color polygons across all the corners.
3. Create multiple nxtgrd corners for conductors on different masks or unspecified colors.

Conductor Layer Thickness Variation

Conductor thickness variation can change circuit behavior dramatically. The extent of the variation depends on the technology node, foundry, and fabrication processes. For example, chemical-mechanical polishing (CMP) is a process technique in which a wafer is physically polished to partially remove deposited layers. CMP processes produce surfaces that are globally flat but that exhibit complex local thickness variations.

The dishing effect is a phenomenon in which the center of a feature is thinner than the edges. The change in conductor cross section affects the resistance and capacitance of the structure. The amount of dishing is dependent on the density of a layer, the spacing from a feature to its neighbors, and the feature width.

Feedback

The StarRC tool performs extraction on designs with thickness variation by first determining the thickness variation for a feature, then computing resistances and capacitances based on the thickness variation.

You can model the process variation by specifying one or more of the following effects in the ITF file:

- The variation of thickness with density
- The weighting factors for different density boxes
- The variation of thickness with width and spacing of conductors
- The orders of density and width for modeling thickness variation using a polynomial equation and the coefficients of the polynomial equation

Single-Box Method

In this method, you choose a single box size and specify the variation of thickness of the conductor in a table. The box is always a square. The maximum size of the box is 500 microns. This method is simple and does not require an exhaustive characterization like the multiple box method. If specified alone for a conductor in the process file, then it does not model local density thickness variation. For linear table modeling, specify a multipoint thickness variation versus density table in the process file.

The `THICKNESS_VS_DENSITY` statement uses the following syntax:

```
THICKNESS_VS_DENSITY [ RESISTIVE_ONLY | CAPACITIVE_ONLY ]
  { (D1 R1) (D2 R2) (D3 R3) (D4 R4) … }
```

D1, D2, D3, D4 represent the density values. *R1, R2, R3, R4* represent the relative change in thickness. Negative R indicates a decrease in thickness and vice versa. Even though R can be any number between -1 and 1, a number close to 1 or -1 is unrealistic. R cannot be –1.

The `THICKNESS_VS_DENSITY` variation affects both resistance and capacitance. However, if the coefficients for resistance and capacitance are different, then use the `RESISTIVE_ONLY` and `CAPACITIVE_ONLY` options.

If no `DENSITY_BOX_WEIGHTING_FACTOR` is specified, the default density box size of 50 microns is used with a weighting factor of unity.

The following combinations are not supported:

- A `THICKNESS_VS_DENSITY` table cannot be combined with `THICKNESS_VS_DENSITY RESISTIVE_ONLY` or `THICKNESS_VS_DENSITY CAPACITIVE_ONLY`.
- The `DENSITY_BOX_WEIGHTING_FACTOR` statement cannot be used without a `THICKNESS_VS_DENSITY` table.

Feedback

The following example illustrates the use of the THICKNESS_VS_DENSITY statement:

```
CONDUCTOR metal3 {
THICKNESS_VS_DENSITY {
(0.1, -0.1) (0.2 0.1) (0.3 0.2) (0.4 0.3)} THICKNESS=0.5
SMIN=0.2 WMIN=0.22 RPSQ=0.06  }
```

Multiple-Box Method

In this method you specify the multiple-box size and its weighting factor for effective density calculation. This method requires that you characterize the wafer in greater detail than the previous method. This method is preferred when the single-box method does not reflect the process behavior. The density box is a square. The maximum size of the density box is 50 microns and the maximum number of boxes is 5.

The following specification is an example that uses four density boxes:

```
THICKNESS_VS_DENSITY { ( D1 R1) (D2 R2) (D3 R3) (D4 R4) }
DENSITY_BOX_WEIGHTING_FACTOR { (S1 W1) (S2 W2) (S3 W3) (S4 W4)}
```

The parameters are as follows:

- S1 to S4 are integers that represent the sizes of the five density boxes in microns. The values must be larger than zero and less than 500 and must be ordered from smallest to largest.
- W1 to W4 are weighting factors. If W is set to 0, then the pair (S W) is ignored. The values must fall between -10 and +10.
- R1 to R4 are the relative thickness values; negative R indicates a decrease in thickness and vice versa.
- D1 to D4 are the density values, or the fraction of the box area occupied by the features. The values must be between 0 and 1.

Calculation of Effective Density

All density calculation is based on drawn width and spacing. When multiple density boxes are specified, the effective density is calculated as shown in Figure 137.

Feedback

Figure 137 Effective Density Calculation

$$D_{eff} = \sum_{i=0 \text{ to } i=5} D(X_i)*W(X_i)$$

D(Xi) - Density of box Xi

W(Xi) - Weighting factor of box Xi

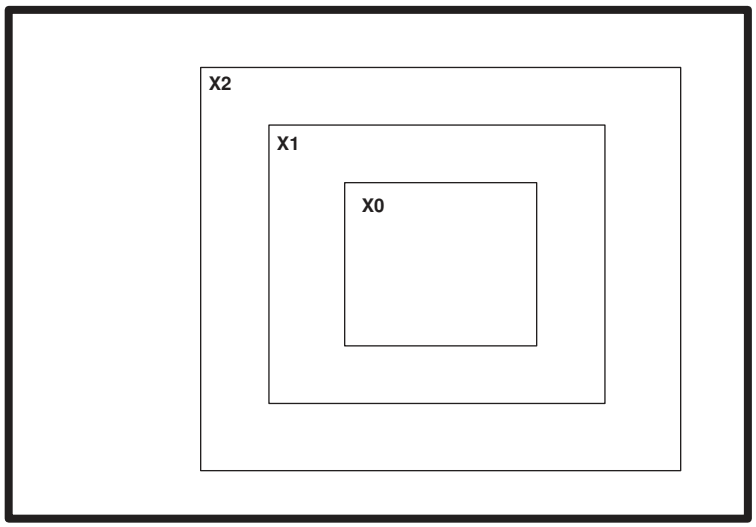


Figure 137 shows three boxes: X0, X1, and X2. StarRC calculates the density for each box for a given segment. The effective density is computed as shown in the equation in Figure 137. After computing the effective density, the variation in thickness is computed based on the `THICKNESS_VS_DENSITY` table. Both resistance and capacitance for a given segment are calculated after thickness modification is taken into account.

Make sure to choose a weighting factor in such a way that calculated effective density is less than unity.

If the computed density exceeds the limit of the density table, the closest density value is picked to calculate the thickness variation.

The following is an example:

```
CONDUCTOR METAL3 {
    THICKNESS = 0.35
    WMIN = 0.2
    SMIN = 0.21
    DENSITY_BOX_WEIGHTING_FACTOR {
        (10, 1) (30, 0.23) (20, 0.29)
        (40, 0.18) (50, -0.12 ) }
    THICKNESS_VS_DENSITY { (0.1, -0.1) (0.2 0.1)}
}
```

Feedback

Width and Spacing-Dependent Thickness Variation

In this method, the variation of thickness as a function of the width of a conductor and the relative spacing to its neighbor is modeled. This thickness variation can be either negative or positive. As can be noted, this is a very local phenomenon and is independent of the density box. If specified with either single or multiple boxes, this thickness variation is computed independently of the density box.

The effective thickness is calculated with the following equation:

$$T = Tnom \times (1 + RTf(Deff) + RTf(W,S) + RTf(SiW))$$

where

- Tnom is the nominal thickness specified in the ITF file.
- RTf(Deff) is the relative change in thickness based on density.
- RTf(W,S) is the relative change in thickness based on width and spacing.
- RTf(SiW) is the relative change in thickness based on silicon width.

The resistance and capacitance is computed after effective thickness is computed. You can model this variation in a process file with the `THICKNESS_VS_WIDTH_AND_SPACING` statement for a conductor.

The `THICKNESS_VS_WIDTH_AND_SPACING` variation affects both resistance and capacitance. However, if the coefficients for resistance and capacitance are different, use the `RESISTIVE_ONLY` and `CAPACITIVE_ONLY` options.

Bottom Conductor Thickness Variation

The bottom conductor thickness variation, or microloading effect, is caused by variation in the trench depth etching process for thin lines. Microloading effects increase as wires become thinner and more closely spaced. Trench depth variation affects the thickness of the interconnect and the separation between metal layers, so it affects both resistance and capacitance.

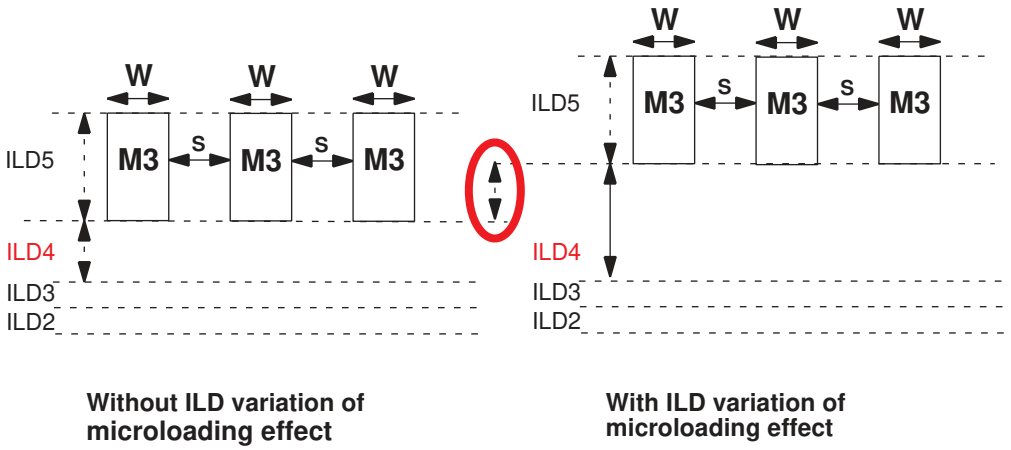
Modeling of the microloading effect can be done in different ways, depending on the data available from foundries.

The `BOTTOM_THICKNESS_VS_SI_WIDTH` statement is used to model microloading as a function of silicon width after etch. The `ILD_VS_WIDTH_AND_SPACING` statement is different from the `BOTTOM_THICKNESS_VS_SI_WIDTH` statement because it models intralayer dielectric (ILD) variation as a function of drawn conductor width and spacing, as opposed to fabricated width and spacing.

Feedback

Figure 138 shows the ILD4 layer thickness varying as a result of microloading on the metal3 conductor.

Figure 138 Model Microloading in the Form of ILD



When the `BOTTOM_THICKNESS_VS_SI_WIDTH` statement is used, the thickness of the conductor can change (increase or decrease). In contrast, when the `ILD_VS_WIDTH_AND_SPACING` statement is used, the conductor thickness remains constant, but the location of the conductor moves up or down, depending on the direction of dielectric variation. This difference might have a significant effect on coupling capacitance to neighboring conductors.

The following restrictions apply to the ILD variation function. Errors are reported to standard output on the terminal screen if any of the following conditions occur:

- The ILD variation is specified for a dielectric layer that does not exist directly below a conductor.
- The ILD variation specified is greater than 0.2 or less than -0.2.
- The ILD variation table is specified in the same `CONDUCTOR` block as a `BOTTOM_THICKNESS_VS_SI_WIDTH` table.

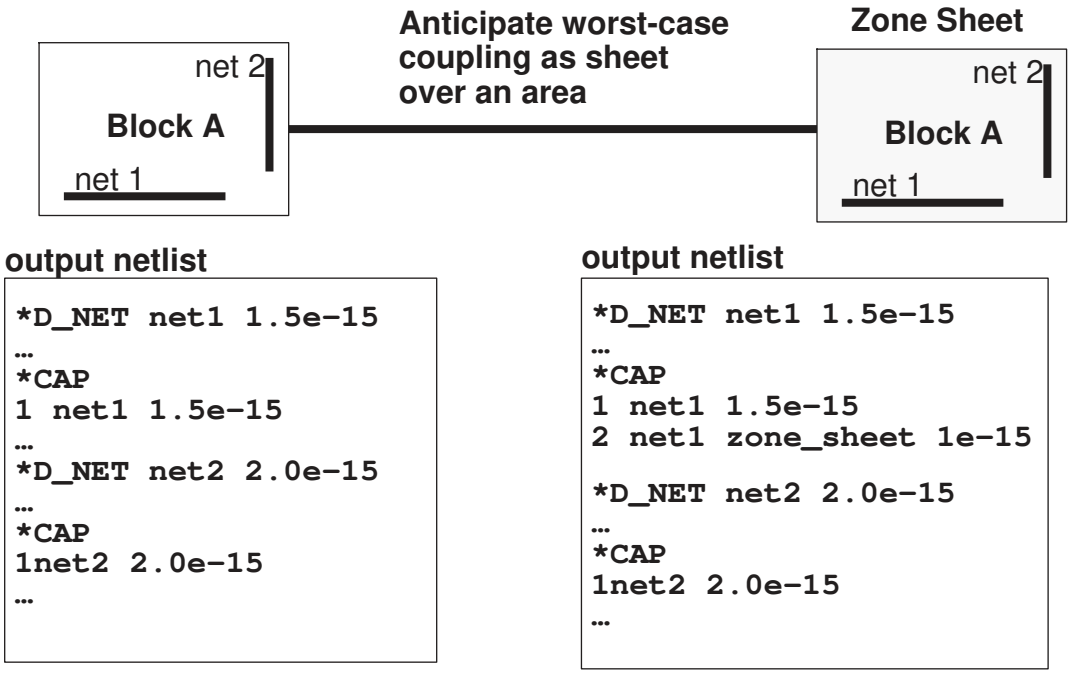
Conductor Sheet Zones

A sheet zone is a location in which you model the insertion of a metal sheet over a specific area as shown in Figure 139. This allows you to measure the coupling capacitance of a given metal sheet, which has a user-defined net name. You can also provide a suffix to the netname. By using sheet zone modeling, you can either specify one sheet or many thin strips of metal with this same command interface.

Feedback

You must ensure that any added sheet zone resides in an area that does not cause metal shorts.

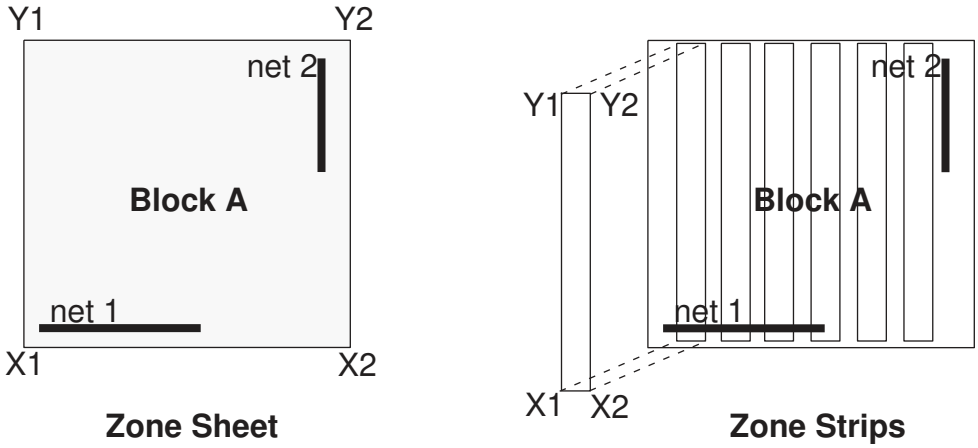
Figure 139 Sheet Zone Modeling



Specify the METAL_SHEET_OVER_AREA command in the command file followed by the metal layer name and four coordinates. These coordinates pinpoint the x- and y-locations of a single sheet as shown in Figure 140. Then specify the SHEET_COUPLE_TO_NET command to designate a unique net name or name prefix. You have the option to specify the SHEET_COUPLE_TO_NET_LEVEL command, which enables a layer-level number to be output as the net name suffix in the output netlist.

Feedback

Figure 140 Specifying a Sheet Zone or Sheet Strips



The following example shows the order of the commands for a single zone sheet:

```
METAL_SHEET_OVER_AREA METAL2 0 0 100 100
METAL_SHEET_OVER_AREA METAL2 200 200 400 400
METAL_SHEET_OVER_AREA METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL:YES
```

The following example shows the order of the commands for several sheet strips:

```
METAL_SHEET_OVER_AREA METAL2 0 5 10 10
METAL_SHEET_OVER_AREA METAL2 8 13 10 10
METAL_SHEET_OVER_AREA METAL2 16 21 10 10
METAL_SHEET_OVER_AREA METAL2 23 28 10 10
METAL_SHEET_OVER_AREA METAL2 31 36 10 10
METAL_SHEET_OVER_AREA METAL2 38 43 10 10
SHEET_COUPLE_TO_NET:zone_strips
SHEET_COUPLE_TO_NET:YES
```

The following limitations accompany the metal sheet capability:

- You must verify that the metal sheet zones you specify do not cause a short.
- The prefix or root net name specified with the `SHEET_COUPLE_TO_NET` command must be unique.

Feedback

Tall Contact Modeling

The ITF file provides several methods to model via layers with high aspect ratios. Tall contacts connect device-level layers such as diffusion or polysilicon layers to higher-level metal layers.

The following sections describe the available modeling methods:

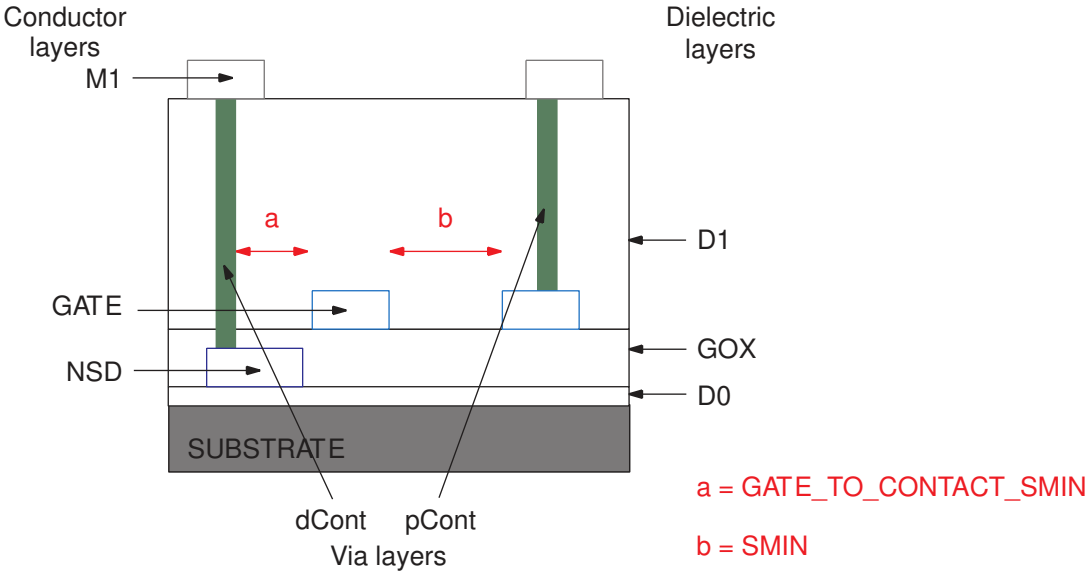
- Standard Tall Contacts
- Tapered Tall Contacts
- Table-Based Modeling of Tall Contacts

Standard Tall Contacts

A simple model might be sufficient for tall contacts that do not have strongly tapered sides and do not exhibit significant coupling capacitance to other tall contacts. For example, in Figure 141, vias pCont and dCont are tall contacts. Write the ITF file as follows:

- Use the `VIA` statement to define the properties of the tall contact via layer.
- Include the `LAYER_TYPE=GATE` option in the `CONDUCTOR` definition for a polysilicon layer in close proximity to the contact.
- Use the `GATE_TO_CONTACT_SMIN` option in the `CONDUCTOR` definition for the polysilicon layer to specify the distance between the polysilicon gate and the tall contact.

Figure 141 Tall Contacts



Feedback

In the ITF file for this structure, the conductor definition contains both the SMIN and GATE_TO_CONTACT_SMIN statements as well as the LAYER_TYPE=GATE specification.

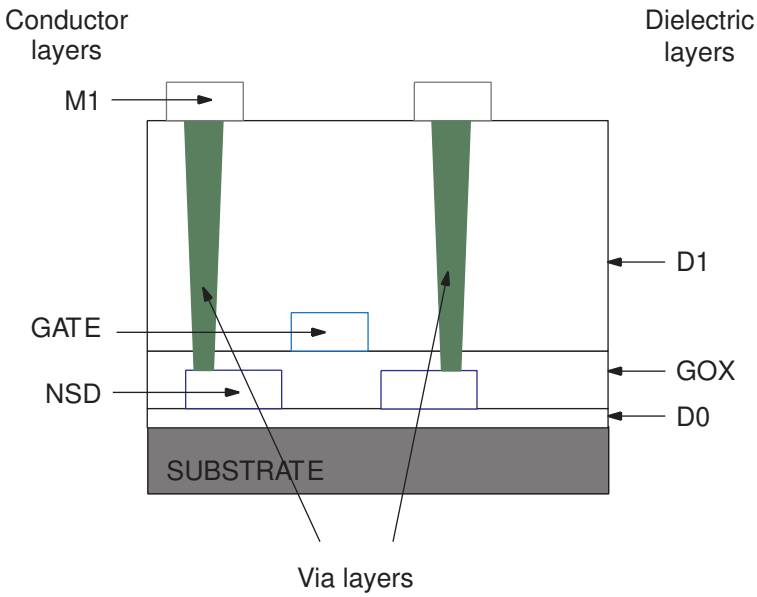
```
TECHNOLOGY = xtor
CONDUCTOR M1 {THICKNESS=0.50 WMIN=0.50 SMIN=0.45 RPSQ=0.062}
DIELECTRIC D1 {THICKNESS=1.8 ER=3.9}
CONDUCTOR GATE {THICKNESS=0.25 LAYER_TYPE=GATE WMIN=0.35
     GATE_TO_CONTACT_SMIN=0.40 SMIN=0.90 RPSQ=3.200}
DIELECTRIC GOX {THICKNESS=0.48 ER=5.0}
CONDUCTOR NSD {THICKNESS=0.40 WMIN=1.00 SMIN=0.35 RPSQ=10.00}
DIELECTRIC D0 {THICKNESS=0.50 ER=3.9}
VIA pCont {FROM=GATE TO=M1 RHO=0.352}
VIA dCont {FROM=NSD TO=M1 RHO=0.500}
```

Tapered Tall Contacts

Tapered contacts have sloped sides, usually wider at the top than at the bottom, as shown in Figure 142. The shape of the contact affects the capacitance between the gate and the contact and the coupling capacitance between adjacent contacts.

To model a simple tapered contact in the ITF file, include the SIDE_TANGENT option in the VIA block for the affected vias. The sidewall angle must be the same for the x- and y-directions. The bottom layer must be either a diffusion layer or a substrate layer. The top layer must be the first metal layer above the gate layer.

Figure 142 Tapered Tall Contacts



Feedback

Table-Based Modeling of Tall Contacts

You can model tall contacts that are very large or that exhibit strong coupling capacitance to other tall contacts by using a table-based method.

Note:

This method cannot be used with the dual polysilicon process.

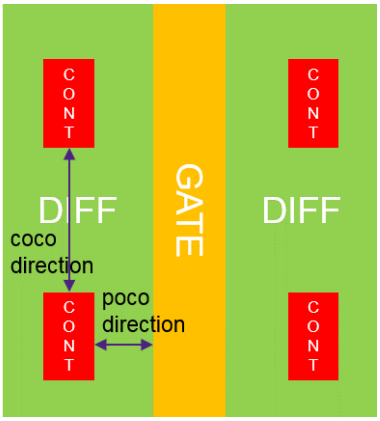
Use the following guidelines to model tall contacts. Figure 144 illustrates the parameters that must be specified in the ITF file.

- For the contact layer, include the following options in the `VIA` definition:
  - Set the `LAYER_TYPE` option to `TALL_CONTACT`.
  - (Optional) Set the `DEVICE_TYPE` option to a unique name. This name should be specified for one via layer and one gate conductor layer to indicate that they are used together in a tall contact structure.
  - (Optional) Include the `CONTACT_WIDTH_AND_LENGTH` option. The argument is a list of pairs of contact width and length values, specified in ascending order. You must specify at least two value pairs, which represent the minimum and maximum contact sizes. You can specify up to nine value pairs to represent typical contact geometries.
  - (Optional) Include the `CONTACT_TO_CONTACT_SPACING` option. The argument is a list of typical spacings between adjacent tall contacts of this type, specified in ascending order. You must specify at least two values, which represent the minimum and maximum values, but no more than five values.

Feedback

- (Optional) Specify the SIDE_TANGENT option with either one or two tangent values to represent sidewall slope, as follows:
  - A single tangent value represents the sidewall angle for both the x- and y-directions.
  - Two tangent values represent separate sidewall angles for the direction between contacts (contact-to-contact or coco direction) and the direction between a contact and the polysilicon or gate shape (poly-to-contact or poco direction). The directions are shown in Figure 143.

Figure 143 Directions for Side Tangent Values

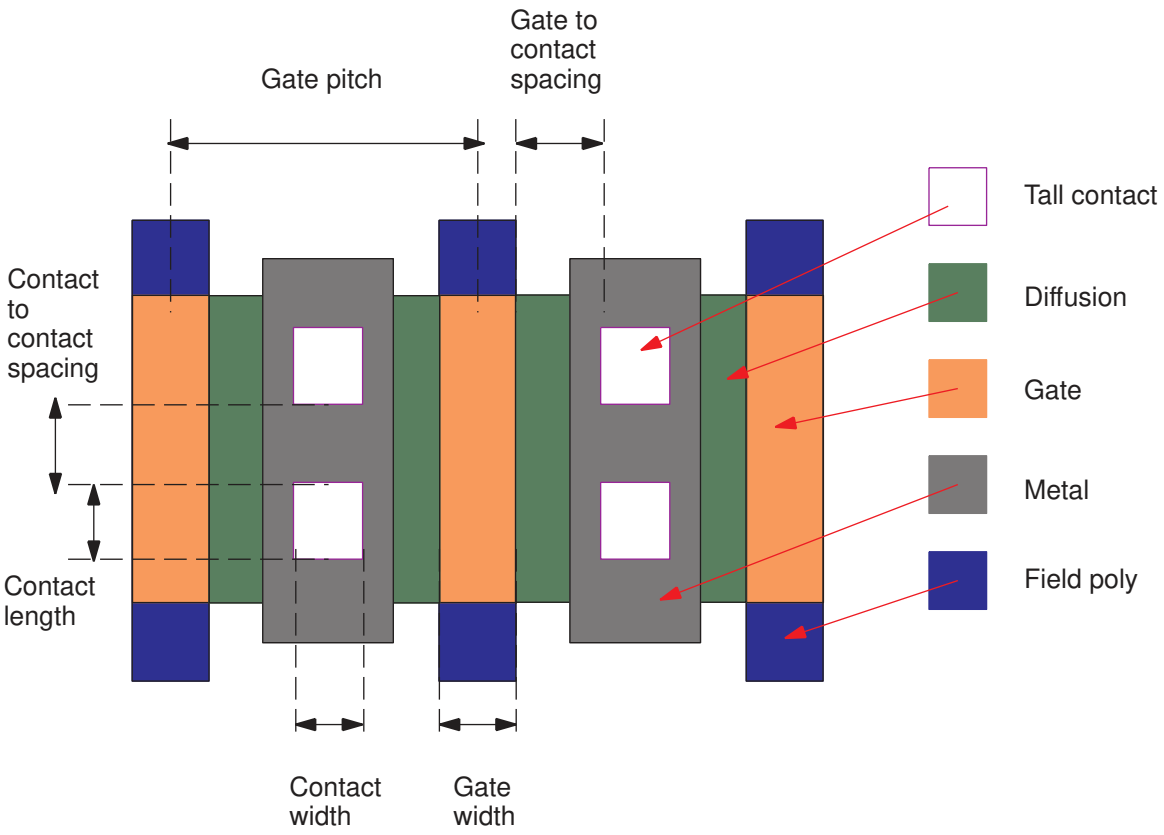


- For the gate conductor layer, include the following options in the CONDUCTOR definition:
  - Set the LAYER_TYPE option to GATE.
  - (Optional) Set the DEVICE_TYPE option to the same name specified in the VIA definition.
  - (Optional) Include the GATE_PITCH option. The argument is a list of typical gate pitch values specified in ascending order. You must specify at least two pitch values, which represent the minimum and maximum values, but no more than three values.
  - (Optional) Include the GATE_WIDTH option. The argument is a list of typical gate width values specified in ascending order. You must specify at least two widths, which represent the minimum and maximum values, but no more than five values.
  - (Optional) Include the GATE_TO_CONTACT_SPACING option. The argument is a list of spacing values between the tall contact and the gate, specified in ascending order. You must specify at least two values, which represent the minimum and maximum spacing, but no more than five spacing values.
- For the diffusion conductor layer, set the LAYER_TYPE option to DIFFUSION.

Feedback

In simultaneous multicorner (SMC) extraction, tall contact definitions must be the same for all corners.

Figure 144 Top View of Tall Contact Layout



Tall Via Modeling

Tall vias connect two metal layers and have high aspect ratios, as shown in Figure 145. They might also have sloped sidewalls. To define tall vias, use the `TALL_VIA_CONFIG` option within a `VIA` block.

The `TALL_VIA_CONFIG` option contains a set of via configuration definitions, which are composed of the following components:

- A name for the via configuration
- X- and y-dimensions for the nominal or drawn (`NOMINAL`) via size, which identifies this via configuration in the layout. A nominal via size can only be used one time in a `TALL_VIA_CONFIG` block.

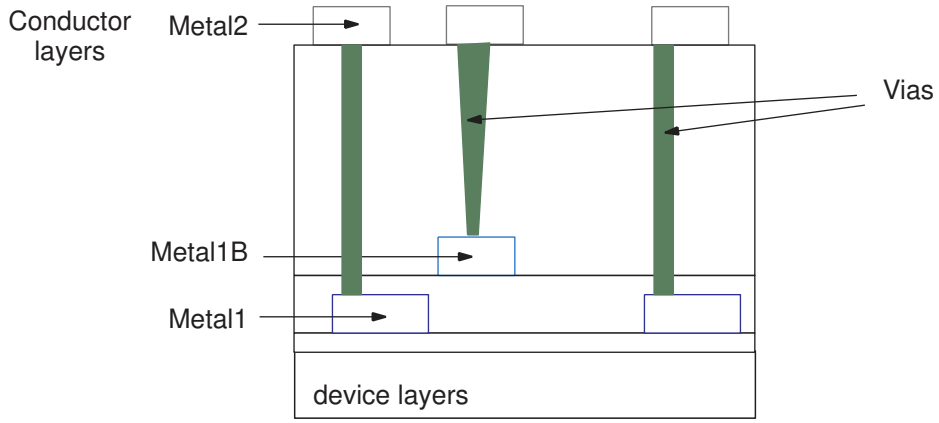
Feedback

All via sizes in the layout should be represented with a via configuration. Unmatched vias have lower extraction accuracy and therefore cause the StarRC tool to issue a warning message.

- X- and y-dimensions for the top of the via (`TOP`) and the bottom of the via (`BOTTOM`). You can specify sloped sidewalls by making the top and bottom dimensions of the via different. The x-dimension and y-dimension can have different slopes.
- A list of named via configurations that might be capacitively coupled to this via configuration (`PAIR_TO`). This list usually includes the via configuration itself and other via configurations defined within the same `TALL_VIA_CONFIG` block. The list can also contain via configurations defined in other `TALL_VIA_CONFIG` blocks. It is not necessary to specify a pair of via configurations under both via names.

  Via configurations paired in this manner are analyzed with increased accuracy at the cost of increased time required to generate the nxtgrd file. Coupling capacitances that are not associated with a `PAIR_TO` keyword are analyzed using standard pattern-based extraction.

Figure 145 Tall Vias



An example of the `TALL_VIA_CONFIG` option is as follows:

```
VIA Via1 {FROM=Metal1 TO=Metal2  RPV=5  AREA=0.01
   TALL_VIA_CONFIG {
      VIA1_A{TOP {0.2 0.2} BOTTOM {0.1 0.1} NOMINAL {0.15 0.15}
         PAIR_TO {VIA1_A VIA0_C}}
      VIA1_B{TOP {0.25 0.15} BOTTOM {0.1 0.07} NOMINAL {0.18 0.12}
         PAIR_TO {VIA1_B VIA1_X}}
      …
      VIA1_X{TOP {0.3 0.3} BOTTOM {0.3 0.3} NOMINAL {0.3 0.3}
         PAIR_TO {VIA1_X VIA1_B}}
```

Feedback

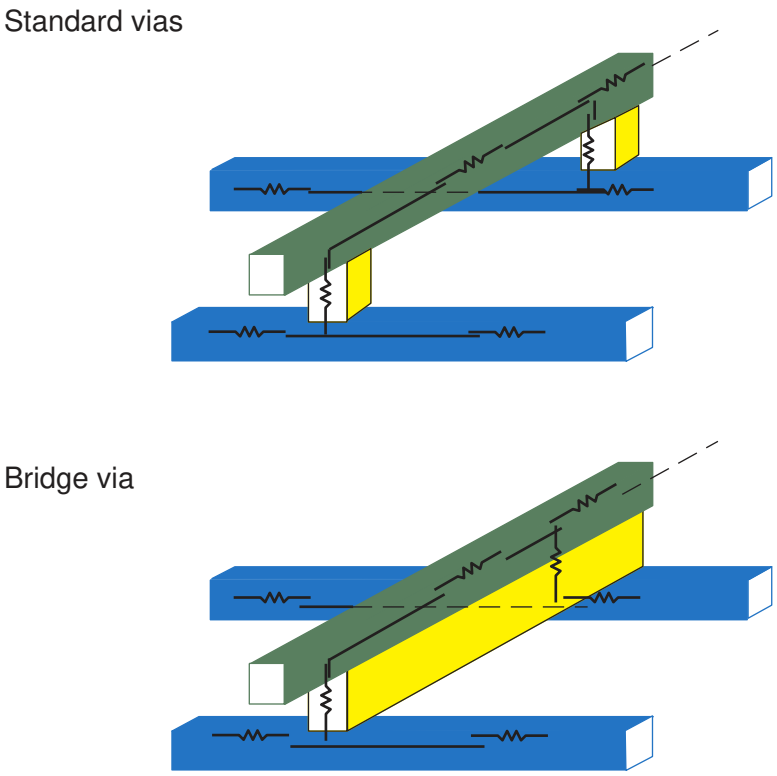
```
        }
    }
```

Bridge Via Modeling

Conventional vias consist of simple connections between two layers. In advanced processing technologies, the via layer sometimes crosses multiple metal lines in either the top or bottom conductor layers. The top diagram in Figure 146 illustrates conventional vias between one metal line in the upper layer and two metal lines in the lower level. The bottom diagram shows a bridge via that connects the same metal lines.

The StarRC tool extracts bridge vias without any changes to the command file. The tool models a bridge via as a set of single vias, as shown by the resistor symbols in Figure 146.

Figure 146 Bridge Via Extraction



Feedback

The following design requirements apply:

- A bridge via can cross two metal lines in either the upper or lower conductor layer. The via cannot cross multiple lines in both layers.
- A bridge via cannot be used only to connect multiple lines on the same conductor layer. The via must connect an upper layer to a lower layer.

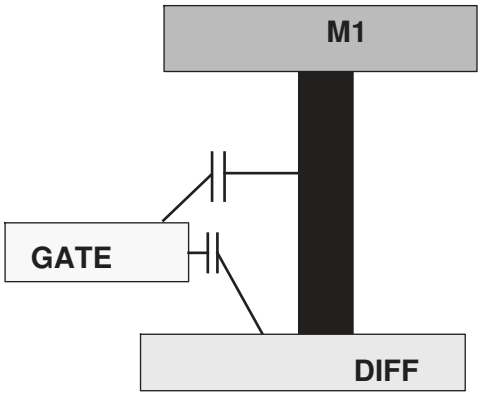
StarRC commands affect the analysis and reporting of bridge vias as follows:

- If the command file contains the `RPV_VS_AREA` command, the entire bridge via area is used to determine the total via resistance. The resistance is divided by the number of connected metal lines to determine the value assigned to each parasitic resistor.
- If the command file contains the `NETLIST_TAIL_COMMENTS: YES` command, the reported via area is the original bridge via area. Via resistors from the same bridge via have the same via layer.
- If the command file contains the `EXTRA_GEOMETRY_INFO: RES` command, the bounding box of the via resistor is the same as the original bridge via.

Gate-To-Diffusion Capacitance Extraction

Parasitic extraction and circuit simulation tools must avoid double counting or elimination of layout-dependent device parasitics, such as gate-to-contact and gate-to-diffusion capacitance, as shown in Figure 147. As process nodes shrink, it is common practice to remove the constant, spatially independent, device-level parasitics from SPICE models in favor of allowing parasitic tools to extract these components.

Figure 147 Layout Dependent Parasitics



This section describes the extraction of the gate-to-diffusion capacitance when the `IGNORE_CAPACITANCE: ALL` command is specified. The gate-to-diffusion

intradevice capacitance is of interest for parasitic extraction tools because of its strong layout dependency. The gate-to-contact capacitance is extracted by using the `EXTRACT_VIA_CAPS: YES` command in the StarRC command file.

To retain the gate-to-diffusion (Cf) capacitance during extraction, use the following command:

```
IGNORE_CAPACITANCE:ALL RETAIN_GATE_DIFFUSION_COUPLING
```

For `IGNORE_CAPACITANCE` settings such as `DIFF` or `NONE`, in which the gate-to-diffusion capacitance is retained by default, StarRC extracts this component as requested.

When you specify this option, the StarRC tool uses the following methods to extract the gate-to-diffusion component:

- Based on precharacterized models, similar to other capacitances
- Based on a 2-D capacitance table look-up dependent on layout parameters

Gate Conductor Resistance

When extracting the resistance of a gate conductor, the StarRC tool considers the full dimensions of the gate polygon to obtain the appropriate value of the resistance per square (using the RPSQ* commands) or resistivity (using the RHO* commands) to calculate the resistance of the gate polygon.

However, the gate resistance is not reported as a single resistance with this value. Instead, the tool reports several smaller resistances that connect to virtual nodes to accurately represent the way that the gate resistance affects the circuit.

The StarRC tool can report the gate resistance in several ways, depending on settings in the command file. In these descriptions, total resistance refers to the resistance calculated for the full gate polygon.

- The standard model reports two resistances in series whose values are each one-half the total resistance. The reported length of each of these resistors is one-half the length of the gate polygon.
- The delta model reports two resistances in series whose values are each one-sixth of the total resistance, along with a negative resistance in parallel whose value is one-half of the total resistance.

  Select this model by using the `MOS_GATE_DELTA_RESISTANCE` command. The reported length of each of the Rg/6 resistors is one-half of the gate polygon length; their widths are equal to the gate polygon width. The reported length of the -Rg/2 resistor is equal to the gate polygon length; its width is fixed at 50 as a flag that it is a nonphysical resistor.

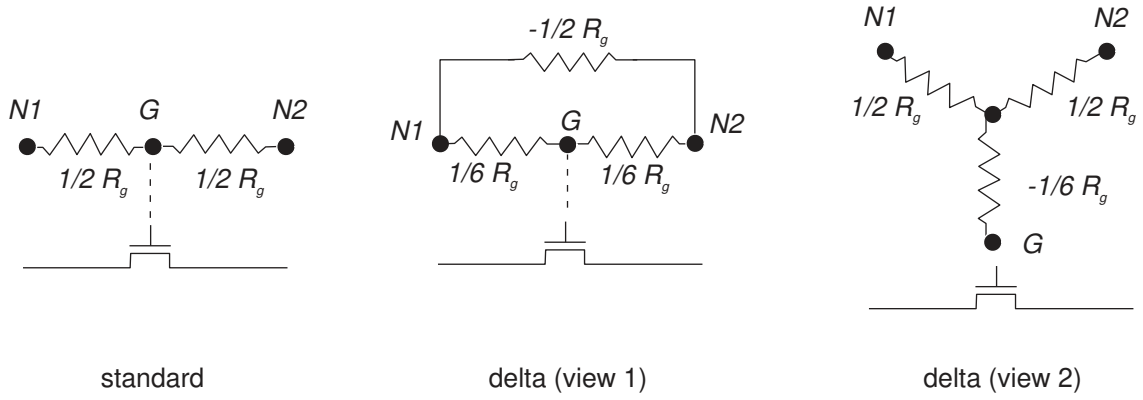
Feedback

- The nonnegative delta model is a modification of the delta model that does not include any negative resistances.

  Select this model by using the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE command. The reported length of each of the the Rg/2 resistors is one-half of the gate polygon length; their widths are equal to the gate polygon width.

Figure 148 shows the standard model and two equivalent views of the delta model. Nodes N1 and N2 represent the ends of the gate polygon, while node G represents the ideal gate terminal location. For the configuration of the nonnegative delta model, see the reference page for the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE command.

Figure 148 Standard Gate Resistance and Two Views of Delta Resistance Network



Conformal Dielectrics

The MEASURED_FROM statement provides the ability to customize the model to account for such process characteristics as conformal dielectrics, mixed conformal and planar dielectrics, and covertical conductors. When used with a DIELECTRIC layer definition, the MEASURED_FROM keyword can either refer to a lower dielectric or can have the value TOP_OF_CHIP. When used with a CONDUCTOR layer definition, the MEASURED_FROM keyword can refer only to a lower PLANAR dielectric.

The TOP_OF_CHIP keyword facilitates the creation of conformal dielectrics. It creates the bottom plane from the layers already present below the new layer and mimics the topology of the existing base layer, copying any existing nonplanarities to the new layer.

The TOP_OF_CHIP keyword is required only if you are creating a conformal layer whose topology is based on the top of the chip. If you want to create a conformal layer that is on top of an existing conformal dielectric, you can measure either from TOP_OF_CHIP or the existing conformal layer.

A MEASURED_FROM statement in CONDUCTOR definitions must always refer to a planar dielectric.

If you create a layer in a MEASURED_FROM command that refers to a planar layer, the new layer is also planar, regardless of whether or not you define TW_T and SW_T.

To regain layer planarity after a conformal dielectric has been defined, take the following steps when defining the new planarized layer:

1. Use the MEASURED_FROM statement to reference a planar dielectric somewhere lower in the process cross section.
2. Adjust the thickness for the new layer so it is equal to its actual physical thickness plus the thickness of any layer on top of the MEASURED_FROM layer.

If you place another dielectric layer on top of the conformal layer without using the MEASURED_FROM statement to regain planarity, use the SW_T and TW_T keywords to set the sidewall and top wall thickness, because the new layer is also conformal.

Conductor Cutting Through Dielectric

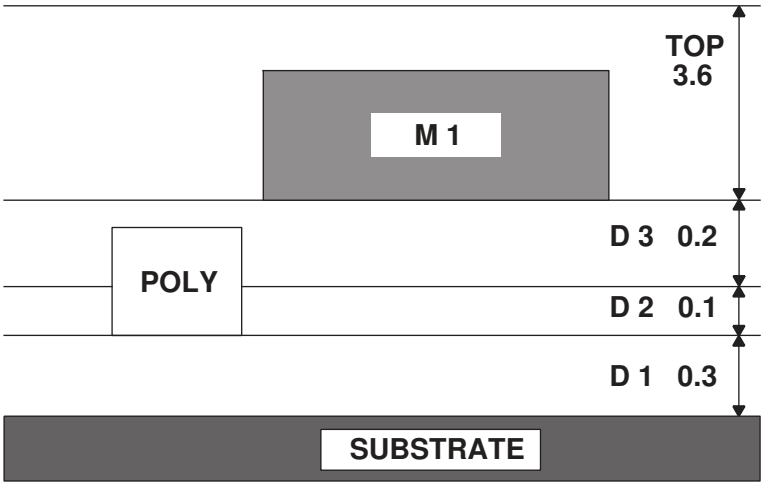
If you use the MEASURED_FROM statement with a conductor and that conductor layer is measured from a dielectric layer that is placed below another dielectric layer, the conductor might cut through the intermediate dielectric. Consider the following example:

```
TECHNOLOGY = SIMPLE
DIELECTRIC TOP { THICKNESS = 3.600 ER = 3.9 }
CONDUCTOR  M1  { THICKNESS = 0.600 WMIN = 0.5
                 SMIN = 0.5 RPSQ = 0.05 }
DIELECTRIC D3  { THICKNESS = 0.300 ER = 3.9 }
CONDUCTOR  POLY{ THICKNESS = 0.200 WMIN = 0.3
                 SMIN = 0.3 RPSQ = 10.0
                 MEASURED_FROM = D1 }
DIELECTRIC D2  { THICKNESS = 0.100 ER = 4.2 }
DIELECTRIC D1  { THICKNESS = 0.300 ER = 3.9 }
```

The process cross section is shown in Figure 149, where POLY cuts through dielectric D2.

Feedback

Figure 149 Conductor Cuts an Intermediate Dielectric



Feedback

Covertical Conductors

The StarRC tool supports covertical (vertically overlapping) conductors.

In this case, the layout database should be modified for the covertical layers, so that those layers (gate and field poly, or poly and local interconnect) do not overlap each other. This can be done in the Hercules runset by use of `BOOLEAN` operations:

```
BOOLEAN POLY NOT LI {} TEMP=POLY
```

or

```
BOOLEAN POLY AND LI {} TEMP=LI_OVERLAP
BOOLEAN POLY NOT LI_OVERLAP {} TEMP=POLY
BOOLEAN LI NOT LI_OVERLAP {} TEMP=LI
```

In the latter case, both `LI` and `LI_OVERLAP` are mapped to the local interconnect layer in the nxtgrd file, and the `CONNECT` sequence in the Hercules runset must be modified accordingly.

Another use for covertical conductors is to handle “metal cheesing” (also known as wide metal slotting); it creates two metal layers and gives them different sheet resistances, which can be done in the mapping file without changing anything in the ITF file, as follows:

```
conducting_layers
  M5 metal5 RPSQ=10
  M5_cheese metal5 RPSQ=100
```

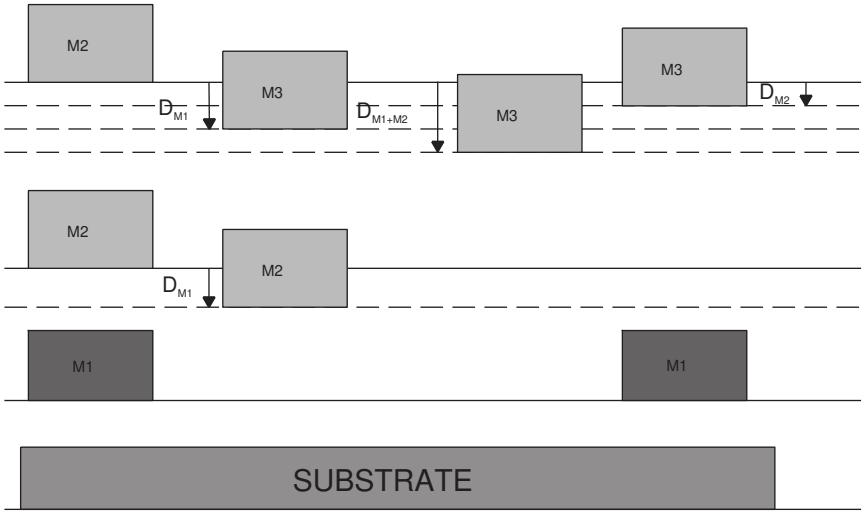
Note that making separate layers in the ITF file for covertical conductors is suitable only for capacitive modeling; you should not use it for modeling resistance differences.

Conductor Drop Factor

The drop factor handles the case in which a conducting layer is at different heights because of the absence of a lower conducting layer. For example, if Metal2 runs over Metal1, Metal2 is uniform at a certain height above Metal1. If Metal2 is layered over a location where there is no Metal1, Metal2 is layered at a lower height. The drop factor considers the differences between the conducting layer heights and calculates the area and lateral capacitance correctly. An illustration of the process is shown in Figure 150.

Feedback

Figure 150 Nonplanar Process With Varying Conductor Heights

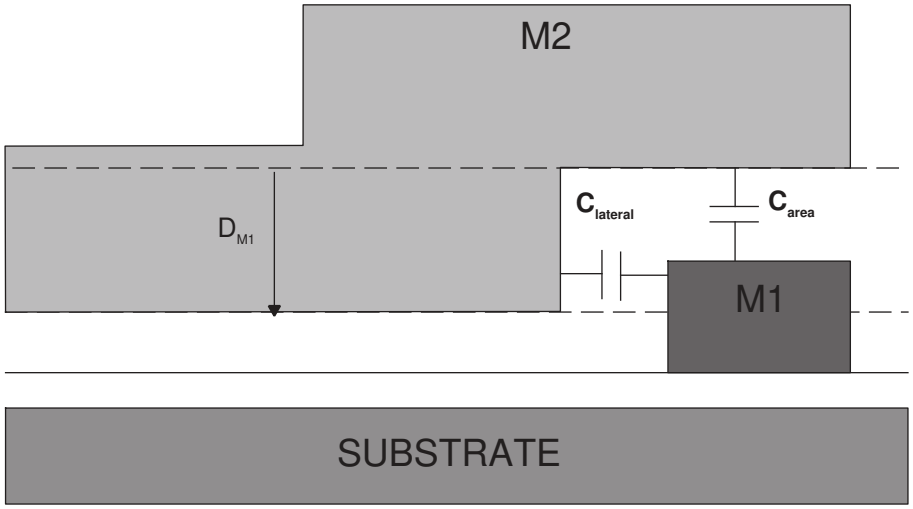


Nonplanar conductor modeling is typically required for legacy processes at process nodes above 0.18 micron with smaller numbers of metal conducting layers, for these reasons:

- Such processes typically contain three metals or less.
- Nonplanarities can be introduced by any missing layer in the physical cross section.
- Both area and lateral capacitance effects are relevant between adjacent metal layers. Depending on the degree of drop of an upper conductor, the drop could introduce a covertical overlap between consecutive conductors that would introduce a potentially significant lateral capacitance effect. For example, see Figure 151.

Feedback

Figure 151 Lateral and Area Capacitance Effects Introduced by Large Drop Factor Values



Drop Factor Error Conditions

The following drop factor usages might result in an error:

- Specifying the `DROP_FACTOR` ITF statement should not cause different horizontally consecutive levels of the same conductor to become noncovertical with each other. In other words, if a piece of conductor routing undergoes a different cumulative drop factor as the number of lower conductors vary along the length of the route, the conductor should never drop such that it can no longer abut with itself. Horizontally adjacent pieces of a conductor can fail to be covertical because of an excessive cumulative drop factor. See Figure 152.
- No conductor can be modeled at a height below conductors represented at lower levels in the ITF cross sectional description. If this is the case, the grdgenxo tool issues an error message. See Figure 153.
- The drop factor is not supported with processes that have these features:
  - Covertical layers such as gate and field polysilicon or polysilicon and local interconnect
  - Metal fill emulation
  - Conformal dielectrics
- You can specify the `DROP_FACTOR` keyword for no more than four conductors.

Feedback

Figure 152 Drop Factor Error Condition 1

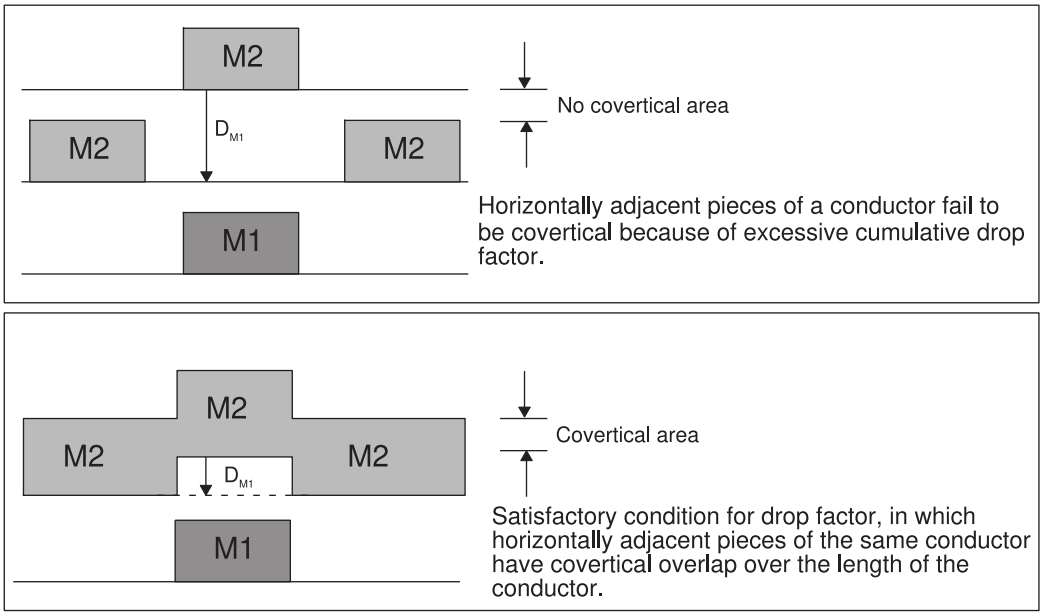
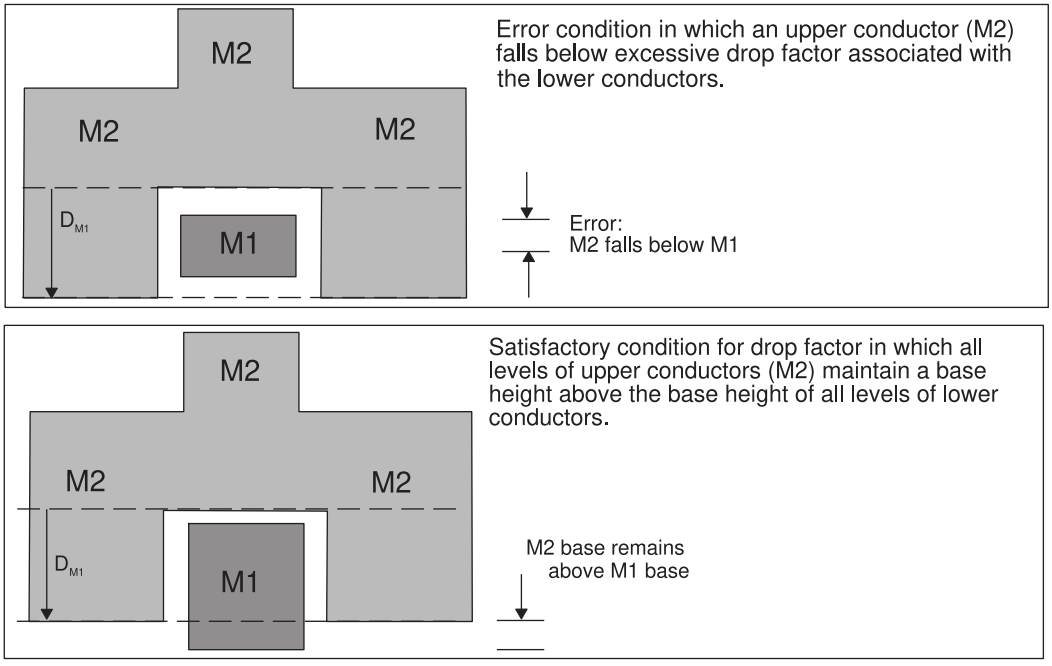


Figure 153 Drop Factor Error Condition 2



Feedback

Dual Polysilicon Gate Process

Some processes use dual polysilicon layers for the transistor gate conductor layer, in conjunction with a tall contact process.

Model a dual polysilicon gate in the ITF file as follows:

- Define the properties of the top polysilicon layer as an ITF conductor layer. Do not include the `LAYER_TYPE` option for this layer.
- Define the properties of the bottom polysilicon layer as another ITF conductor layer and set the `LAYER_TYPE` option to `GATE`. Set the `DUAL_POLY` option to the name of the top polysilicon layer.

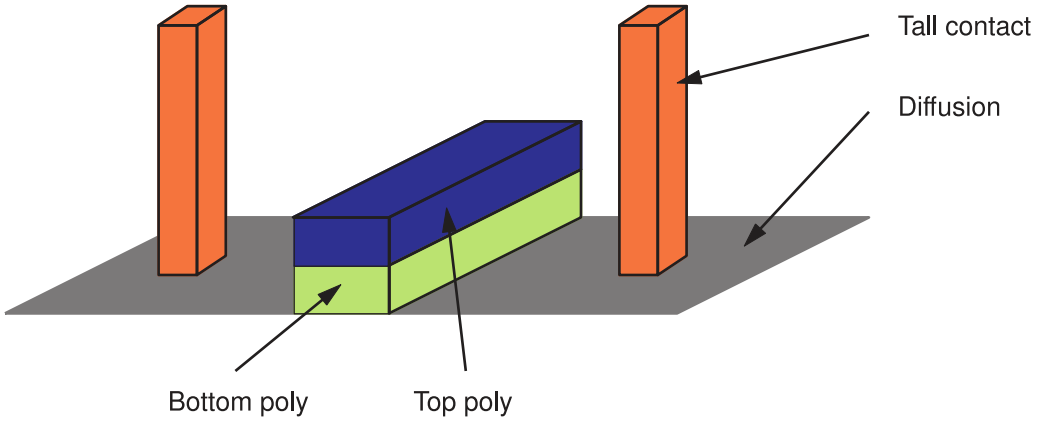
This process is only valid when used with tall contacts. The contact height must be at least ten times the combined thickness of the top and bottom polysilicon layers.

The top and bottom polysilicon layers must meet the following conditions:

- The layers must be separated by a vertical distance of less than 15 Angstroms.
- The layers must have the same etch value or the same set of etch tables in their ITF file definitions.
- The layers must share the same ignore capacitance properties in the `ignore_capacitance` section of the mapping file.
- The layout polygons must have the same width (the dimension along the transistor channel length direction).

Feedback

Figure 154 Dual Polysilicon Gate Transistor With Tall Contacts



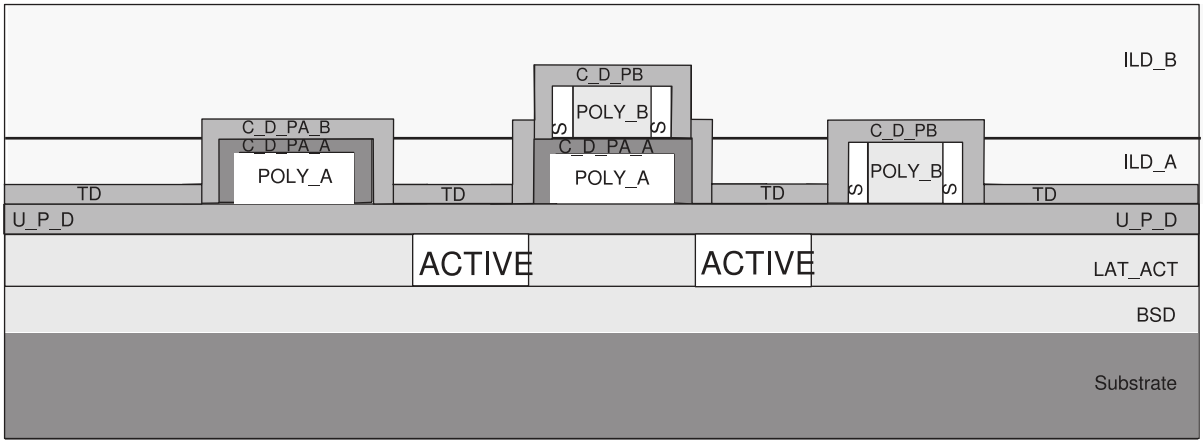
Feedback

Double-Polysilicon Process

To model a double-polysilicon process, set the `IS_CONFORMAL` and `ASSOCIATED_CONDUCTOR` options in the `DIELECTRIC` layer block of the ITF file. The `IS_PLANAR` option is necessary in this case to make the metals above the poly layers planar.

See an example of this cross section in Figure 155.

Figure 155 Conformal Dielectric and Poly Layers



The following ITF statements model the cross section shown in Figure 155.

```
DIELECTRIC ILD_B { THICKNESS=0.3 MEASURED_FROM=ILD_A ER=4.2 }
DIELECTRIC C_D_PB { IS_CONFORMAL ER=7 SW_T=0.03 TW_T=0.03
     ASSOCIATED_CONDUCTOR=POLY_B}
DIELECTRIC S{IS_CONFORMAL ER=6 SW_T=0.055 TW=0.0
     ASSOCIATED_CONDUCTOR=POLY_B}
CONDUCTOR POLY_B { THICKNESS=0.15 WMIN=0.08 SMIN=0.07 RPSQ=10.0 }
DIELECTRIC ILD_A { THICKNESS=0.10 MEASURED_FROM=TD ER=4.2 }
DIELECTRIC TD { ER=7 MEASURED_FROM=U_P_D THICKNESS=0.03 }
DIELECTRIC C_D_PA_B { IS_CONFORMAL ER=7 SW_T=0.03 TW_T=0.03
     ASSOCIATED_CONDUCTOR=POLY_A }
DIELECTRIC C_D_PA_A { IS_CONFORMAL ER=3.9 SW_T=0.04 TW_T=0.01
     ASSOCIATED_CONDUCTOR=POLY_A }
CONDUCTOR POLY_A { THICKNESS=0.12 WMIN=0.05 SMIN=0.05 RQSP=849
     DROP_FACTOR=0.13 }
DIELECTRIC U_P_D { THICKNESS=0.05 ER=3.9 }
DIELECTRIC LAT_ACT { THICKNESS=0.19 ER=4 }
CONDUCTOR ACTIVE { THICKNESS=0.19 WMIN=0.1 SMIN=0.14 RPSQ=0.0001 }
DIELECTRIC BSD { THICKNESS=0.19 ER=4 }
```

Feedback

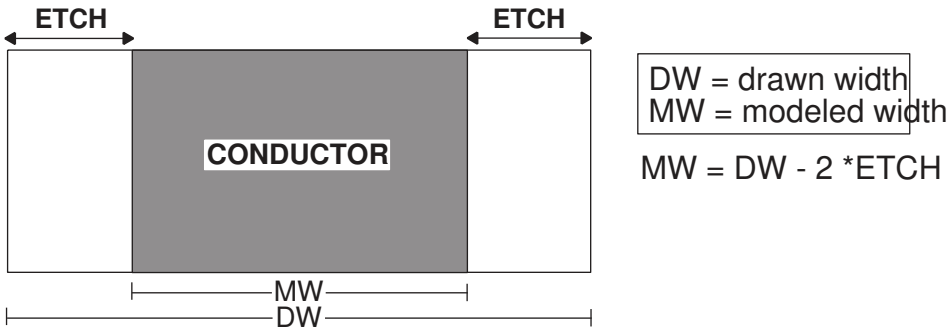
Layer Etch

You can make an adjustment in the ITF process file for layer etch effects that cause the manufactured line width of a conductor to be different from its drawn width. The ETCH statement can be specified as a part of any CONDUCTOR definition.

Both conductor sidewalls retreat or expand by the value specified in the ETCH statement, resulting in a net width change of twice the etch value, as illustrated in Figure 156. A positive etch value shrinks the conductor width, and a negative ETCH value increases the conductor width.

WMIN and SMIN values are not affected by the ETCH statement because the StarRC tool makes the etch adjustments internally.

Figure 156 Process Using Layer Etch Adjustment



See Also

- Layer Etch Process ITF Example

Spacing- and Width-Dependent Etch

Spacing- and width-dependent etch can be implemented in the nxtgrd with the ETCH_VS_WIDTH_AND_SPACING option within a CONDUCTOR block of the ITF file.

With this feature, the StarRC tool considers the fabricated patterns when extracting parasitics. This is important, because optical proximity correction (OPC) cannot fix all proximity effects and the actual patterns might be different from the drawn mask patterns.

If you add the ETCH_VS_WIDTH_AND_SPACING option to an existing ITF file, you must rerun the grdgenxo tool after removing the working directory.

Feedback

The ETCH_VS_WIDTH_AND_SPACING option can be used with the ETCH statement. If these statements are used together, the ETCH_VS_WIDTH_AND_SPACING calculation is applied before the ETCH adjustment, then the RPSQ_VS_SI_WIDTH value is calculated.

The ETCH_VS_WIDTH_AND_SPACING option affects both capacitance and resistance by default. You can use the CAPACITIVE_ONLY or RESISTIVE_ONLY options to restrict the effect to capacitance or resistance, respectively.

Determining WMIN and SMIN Values

It is important to have a correct set of WMIN and SMIN values for the CONDUCTOR object that contains the ETCH_VS_WIDTH_AND_SPACING statement.

The WMIN and SMIN values of the conductor described by the ETCH_VS_WIDTH_AND_SPACING statement can be the same as the smallest value (or the first entry) in the WIDTHS and SPACINGS tables, respectively.

Inappropriate WMIN and SMIN values might cause unwanted opens or shorts of the neighboring layers by applying the etch values provided in the table. In such a case, a message is printed during the run. For the entries corresponding to the WMIN in the WIDTHS table, if positive etch values are greater than or equal to half of the WMIN value, an open warning is issued.

For the entries corresponding to the SMIN value in the SPACINGS table, if absolute values of the negative etch are greater than or equal to half of the SMIN value, a potential short condition exists. However, reporting of this condition is optional because most such errors should be caught during design rule checking. To enable SMIN violation reporting, use the REPORT_SMIN_VIOLATION: YES command in the StarRC command file (the default is NO).

Overlapping Wells

You can set the vertical profile of the substrate. To set the vertical precedence for layers mapped to the substrate, use the following mapping file syntax:

```
conducting_layers
db_layer1    SUBSTRATE    precedence=pos_integer
db_layer2    SUBSTRATE    precedence=pos_integer
…
```

Any layer mapped to the substrate, and only layers mapped to the substrate, accepts a positive integer precedence value that establishes the layer's position in the vertical substrate profile. Larger numbers denote higher vertical precedence, while smaller numbers denote lower vertical precedence. It is not necessary for values to be sequential.

If two layers have the same precedence value, and polygons from those two layers overlap in the layout, the StarRC tool chooses the topmost layer for the purpose of

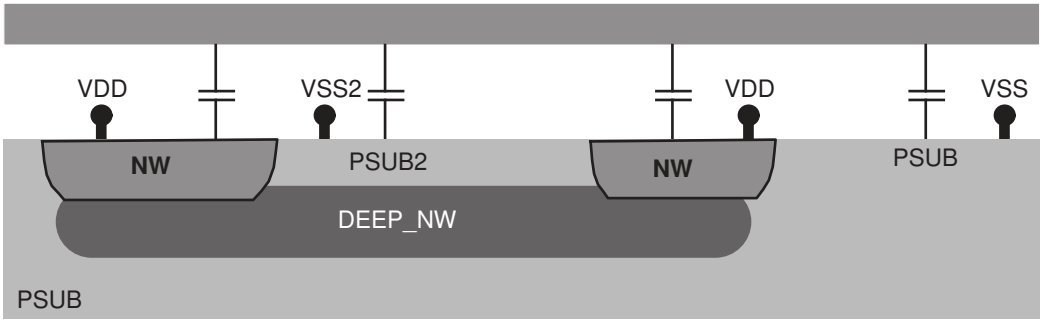
Feedback

coupling capacitance attachment and IGNORE_CAPACITANCE command functionality. SUBSTRATE-mapped layers for which precedence is not specified have a precedence value of zero, meaning that their precedence is lower than all other layers.

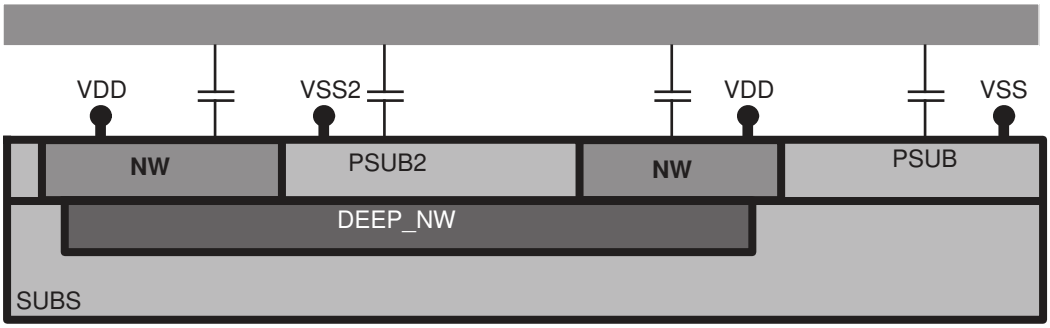
The following mapping file example shows a mapping file used to establish vertical precedence for a buried well profile. Figure 157 shows the profile of a physical well for a buried well process and a profile for a discrete well.

```
conducting_layers
    SUBS      SUBSTRATE  precedence=1
    DEEP_NW   SUBSTRATE  precedence=2
    NW        SUBSTRATE  precedence=3
    PSUB2     SUBSTRATE  precedence=3
    PSUB      SUBSTRATE  precedence=3
```

Figure 157 Physical Well and Discrete Buried Well Profile



Physical well profile for buried well process



Discrete buried well profile for parasitic extraction

Damage Modeling

For advances process nodes, sidewall and bottom wall damage as a consequence of processing low-k dielectrics might need to be considered.

Feedback

The DAMAGE_THICKNESS ITF statement defines the thickness of the damage region on the surface of the layer, while the DAMAGE_ER ITF statement specifies the equivalent permittivity of the damage region.

Figure 158 shows the damage modeling cross sections that are modeled with the DAMAGE_ER and DAMAGE_THICKNESS statements.

Figure 158 Damage Modeling Cross Sections

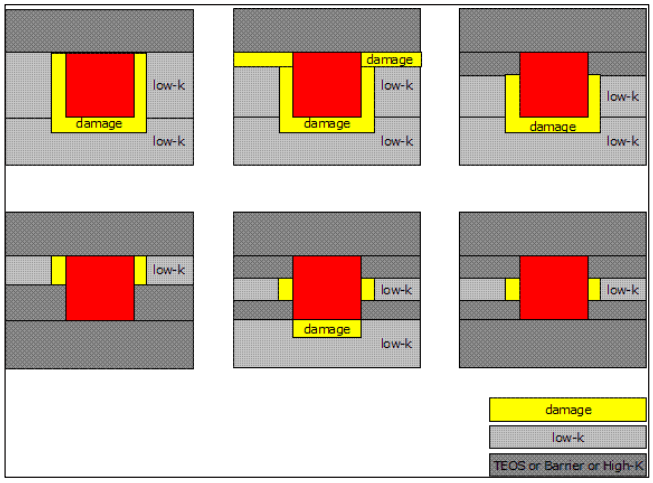
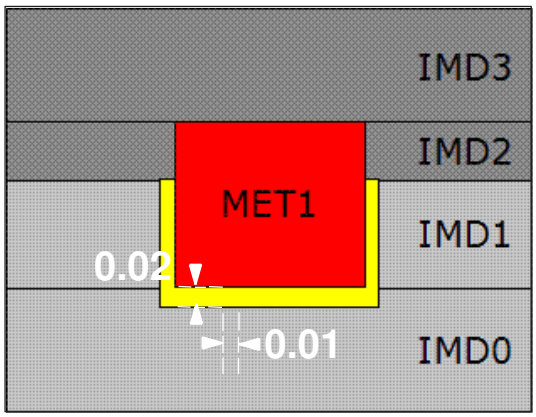


Figure 159 Low-K Damage



In Figure 159, the damage defined for IMD1 models the side wall damage because IMD1 is the intrametal dielectric for metal1, whereas IMD0 models the bottom wall damage because metal1 is on top of the dielectric layer IMD0.

The following example corresponds to Figure 159.

```
DIELECTRIC IMD3 { THICKNESS=0.09 ER=2.9 }
DIELECTRIC IMD2 { THICKNESS=0.07 ER=4.5 }
```

```
DIELECTRIC IMD1 { THICKNESS=0.13 ER=2.9
DAMAGE_THICKNESS=0.01 DAMAGE_ER=5.1 }
    CONDUCTOR  MET1 { THICKNESS 0.20 SMIN=0.1 WMIN=0.1 }
DIELECTRIC IMD0 { THICKNESS=0.10 ER=2.9
    DAMAGE_THICKNESS=0.02 DAMAGE_ER=5.1 }
```

Temperature Derating

The StarRC tool can model temperature-dependent resistance for conducting layers and vias. The resistances are modeled in the same way as they are in SPICE, by using the following equation:

```
R = R0 * [1 + CRT1 * (T - T0) + CRT2 * (T - T0)^2]
```

`R0` is the resistance value at the nominal temperature `T0`, `CRT1` and `CRT2` are linear and quadratic temperature coefficients, and R is the modeled resistance at the operating temperature T. Note that the modeled resistance R exactly equals the nominal resistance (`R0`) if `T=T0`, or if both `CRT1=0` and `CRT2=0`.

The statement options `CRT1`, `CRT2`, and `T0` are specified on a per-layer basis. If either `CRT1` or `CRT2` is nonzero for a layer (vias included), then a nominal temperature is required for that layer.

The defaults for `CRT1` and `CRT2` are zero. The default nominal temperature for the layers can be set globally with the `GLOBAL_TEMPERATURE` statement at the beginning of the ITF file. If the temperature is set both globally and at a layer, the layer nominal temperature overrides the global setting.

```
GLOBAL_TEMPERATURE = temp_in_Celsius
```

Half-Node Scaling

The half-node scaling function allows you to scale the input design data uniformly across all layers without affecting the connectivity of the layout network.

The `HALF_NODE_SCALE_FACTOR` statement allows for a transparent shrink flow. The flow requires downstream tools to interpret this option. This flow produces modified electrical properties, for example resistance and capacitance, but retains the original unshrunk design coordinates for the final netlist. The `HALF_NODE_SCALE_FACTOR` function does not require scaling options to be set in other tools. The technology files supplied to the designer (from the foundry or CAD design group) contain the scaling factor for the particular design flow.

Feedback

How the Function Works

Set the HALF_NODE_SCALE_FACTOR option in the ITF file as shown in the following example:

```
TECHNOLOGY = 65nm_example
GLOBAL_TEMPERATURE = 25
HALF_NODE_SCALE_FACTOR = 0.9
DIELECTRIC PASS2 {THICKNESS=0.800000 ER=6.9}
$DIELECTRIC PASS1 {THICKNESS=0.700000 ER=4.0}
```

If a half-node scale factor exists in the nxtgrd file, the StarRC tool automatically sets the MAGNIFY_DEVICE_PARAMS command to NO and issues a warning message. Table 59 describes the effect of other StarRC commands.

Table 59 Half-Node Scale Factor Effect With StarRC Commands

Command	Function with HALF_NODE_SCALE_FACTOR
MAGNIFY_DEVICE_PARAMS: NO	Writes standard device properties ($w, $l, $area, and so on) at full node values in transistor-level flows (set to NO automatically)
NETLIST_DEVICE_LOCATION_ORIENTATION: YES	Writes full node (original GDSII) coordinates in the netlist for flows requiring device locations
NETLIST_UNSCALED_COORDINATES: YES	Writes coordinate information in the netlist at full node values
NETLIST_UNSCALED_COORDINATES not set	Writes coordinate information in the netlist at full node (set to YES automatically) values
NETLIST_UNSCALED_COORDINATES: NO	Writes coordinate information in the netlist at scaled node values

How Scaling Affects The Netlist

The following is an example of coordinates that are affected in a SPICE-like netlist:

```
*|I (ZN:F12 M1 SRC B 0 0.48 0.64)
*|P (ZN B 0 0.695 1.115)
*|S (ZN:16 0.53 1.545)
```

If the NETLIST_TAIL_COMMENTS command is used to write the physical properties of parasitic resistors (used for reliability analysis flows), the properties are the full-node size:

```
R1 F9 F8 0.588229 $l=9.9 $w=2 $lvl=1
```

Feedback

The HALF_NODE_SCALE_FACTOR option does not change the function of the MAGNIFICATION_FACTOR option. However, you cannot use the MAGNIFICATION_FACTOR option with the HALF_NODE_SCALE_FACTOR option.

Changing a Scale Factor Value in the nxtgrd File

If you generated an nxtgrd file without setting a scale factor, or you would like to change the scale factor, run the grdgenxo tool to generate an updated nxtgrd file. The following example sets the scale factor to 0.9:

```
% grdgenxo -add_sf 0.9 -i noshrink.nxtgrd -o shrink.nxtgrd
```

If you generated an nxtgrd file with a HALF_NODE_SCALE_FACTOR value and you would like to run extraction without the shrink, remove the scaling factor from the nxtgrd file by setting the factor to 1, as follows:

```
% grdgenxo -add_sf 1 -i noshrink.nxtgrd -o shrink.nxtgrd
```

Note:

You can set the MAGNIFICATION_FACTOR command in the command file after changing the value in the nxtgrd file. You cannot delete the HALF_NODE_SCALE_FACTOR line from the nxtgrd file as this causes the nxtgrd file to be corrupt.

Interface to Reliability Flows

Reliability tools read the parasitic netlist. Because the netlist from represents full node coordinates, the reliability tool's electromigration rules are supported at the full node.

The half node scale factor is written as a comment in the final netlist for downstream analysis tools to compute the physical width for estimation. For example:

```
//COMMENTS
//TCAD_GRD_FILE /remote/na4apd/starrc/group/option_2/tcad/grd
//TCAD_TIME_STAMP Sun Feb 18 12:08:22 2007
//TCADGRD_VERSION 56
//HALF_NODE_SCALE_FACTOR 0.9
```

Via Merging

The StarRC tool handles via merging differently for gate-level and transistor-level extraction.

- For gate-level flows, vias are merged by default. To disable via merging, set the MERGE_VIAS_IN_ARRAY command to NO.
- For transistor-level flows, via merging behavior is controlled by settings in the StarRC command file, the ITF file, and the mapping file. In addition, electromigration flows follow different via merging rules.

Feedback

In a via array to be considered for merging, the horizontal spacing between vias must be the same and the vertical spacing between vias must be the same. However, the horizontal spacing can be different from the vertical spacing.

Via Merging in Standard Transistor-Level Flows

For transistor-level flows, via merging is controlled by settings in the StarRC command file, the ITF file, and the mapping file. This topic describes how these settings affect the maximum via spacing, maximum via count, and maximum array length.

Transistor-level via merging applies to both power and signal nets, but not to contacts or virtual vias.

Maximum Via Spacing

The default maximum spacing between adjacent vias in the x and y directions is calculated as follows:

- If the `AREA` option of the `via_layers` mapping file command is specified, the maximum spacing value is twice the square root of the `AREA` value, as follows:

  $$S_{max} = 2 \times \sqrt{Area}$$

- If the `AREA` option of the `via_layers` mapping file command is not specified, but an `RPV_VS_AREA` table exists for the via layer in the ITF file, the tool uses the first area value in the `RPV_VS_AREA` table to calculate S_{max}.

You can customize the maximum allowable spacing between vias by specifying the `MAX_VIA_ARRAY_SPACING` option of the `via_layers` mapping file command.

Maximum Via Array Length or Via Count

The default maximum via array length in the x and y directions is calculated as follows:

- If the `AREA` option of the `via_layers` mapping file command is specified, the maximum spacing value is ten times the square root of the `AREA` value, as follows:

  $$L_{max} = 10 \times \sqrt{Area}$$

- If the `AREA` option of the `via_layers` mapping file command is not specified, but an `RPV_VS_AREA` table exists for the via layer in the ITF file, the tool uses the first area value in the `RPV_VS_AREA` table to calculate L_{max}.

You can customize the maximum allowable via array length by specifying the `MAX_VIA_ARRAY_LENGTH` option of the `via_layers` mapping file command.

Alternatively, you can specify the maximum maximum number of vias to merge on a specific database layer by using the `VIA_ARRAY_COUNT_LAYER` command. You can also

Feedback

provide this information in a separate file specified with the VIA_ARRAY_COUNT_FILE command. If the same layer is specified in both a file and a VIA_ARRAY_COUNT_LAYER command, the VIA_ARRAY_COUNT_LAYER command takes precedence.

For example, if the maximum via count is set to 3, the tool locates a 3x3 array of vias, merges that array into a single via, then examines additional vias in both directions and continues to merge 3x3 arrays into single vias.

Via merging for layers specified in the VIA_ARRAY_COUNT_LAYER or VIA_ARRAY_COUNT_FILE command occurs even if the MERGE_VIAS_IN_ARRAY command is set to NO.

Table 60 Via Merging Behavior for Standard Transistor-Level Flows

VACL (Via Array Count Layer or File) provided	MERGE_VIAS_IN_ARRAY	Mapping file parameters SET (MVAL and MVAS)	Mapping file parameters NOT SET
YES	YES	Max count = value in VACL Max array length = unlimited Max spacing = MVAS if present, otherwise default	Max count = value in VACL Max array length = unlimited Max spacing = default
YES	NO	Max count = value in VACL Max array length = unlimited Max spacing = MVAS if present, otherwise default	Max count = value in VACL Max array length = unlimited Max spacing = default
NO	YES	Max count = unlimited Max length = MVAL if present, otherwise default Max spacing = MVAS if present, otherwise default	Max count = unlimited Max length = default Max spacing = default
NO	NO	Max count = unlimited Max length = MVAL if present, otherwise default Max spacing = MVAS if present, otherwise default	Do not merge vias

The output netlist contains one subnode (*|S) for every resultant via array. The resistors are listed with an effective resistance value including a summation of area for all individual vias in the group. Figure 160 illustrates the effective resistances of three conductors connected by merged via arrays.

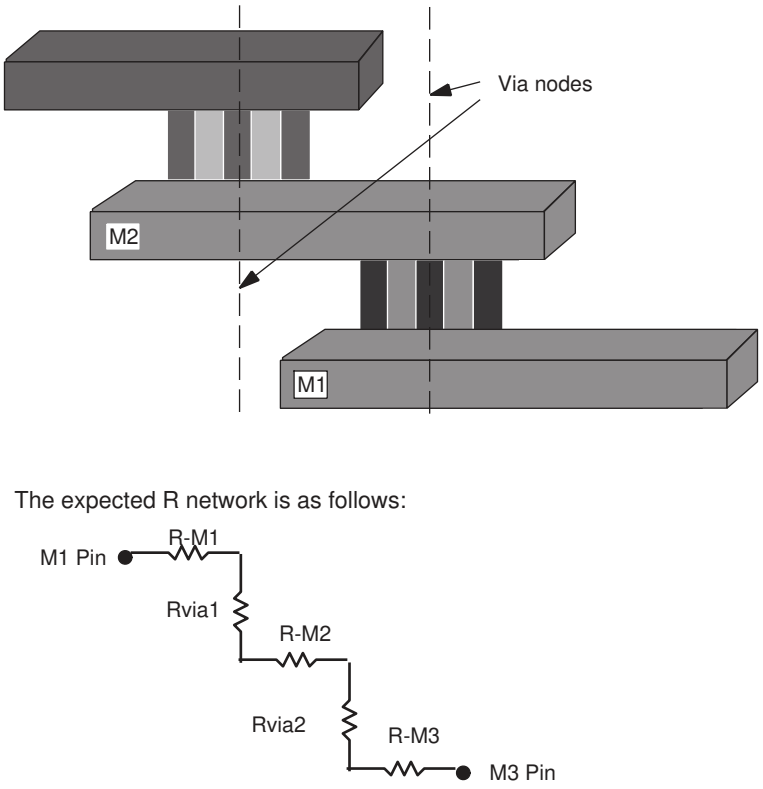
The mapping file syntax is as follows:

```
VIA   GRD_VIA RPV = value
      AREA = value
```

Feedback

```
MAX_VIA_ARRAY_SPACING = value
MAX_VIA_ARRAY_LENGTH = value
```

Figure 160 Resistor Network for a Multilayer Net



Via Merging in Transistor-Level Electromigration Flows

Certain electromigration flows require a limit to the maximum spacing allowed for via merging, as follows:

$$S_{max} = 1.25 \times (W_{min} + S_{min})$$

where S_{max} is the maximum spacing between adjacent vias in the x and y directions, W_{min} is the minimum via width, and S_{min} is the minimum spacing between vias.

Feedback

To enable this limit, you must set W_{min} and S_{min} values, using one of the following methods. If both of these sets of values are specified, the values in the StarRC command file take precedence.

- Specify the `VIA_WMIN` and `VIA_SMIN` commands in the StarRC command file.
- Specify `WMIN` and `SMIN` values for the `VIA` layer in the ITF file.

In the electromigration flow, the tool always performs via merging and ignores the `MERGE_VIAS_IN_ARRAY` command.

Table 61 summarizes the behavior for electromigration flows.

Table 61 Via Merging for Electromigration Flows When Wmin and Smin Provided

VACL (Via Array Count Layer or File provided)	Mapping file parameters SET (MVAL and MVAS)	Mapping file parameters NOT SET
YES	Max count = value in VACL Max array length = unlimited Max spacing = 1.25*(Wmin + Smin), otherwise default	Max count = value in VACL Max array length = unlimited Max spacing = 1.25*(Wmin + Smin), otherwise default
NO	Max count = unlimited Max length = MVAL if present, otherwise default Max spacing = 1.25*(Wmin + Smin), otherwise default	Max count = unlimited Max length = default Max spacing = 1.25*(Wmin + Smin), otherwise default

Feedback

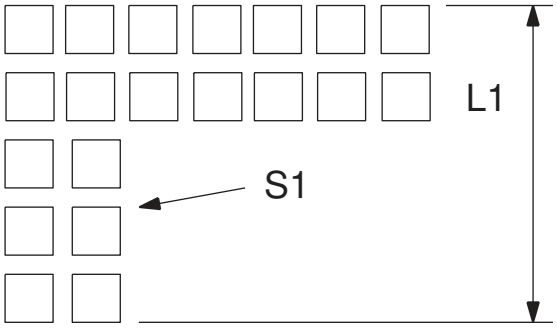
Examples of Via Merging

The following examples demonstrate how the StarRC tool merges vias in different configurations of via arrays.

Grouping Vias in an L-Shaped Array

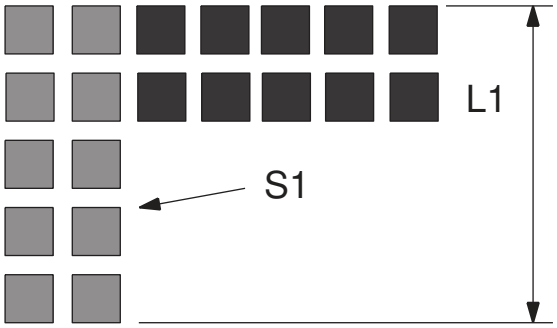
In an L-shaped via array, the StarRC tool groups the vias into multiple sets in an arbitrary manner as shown in Figure 161.

Figure 161 Case 1 - Before Merge



If you specify `MAX_VIA_ARRAY_SPACING = S1` and `MAX_VIA_ARRAY_LENGTH = L1` in the `via_layers` section of the mapping file, the via array might be divided into two groups.

Figure 162 Case 1 - After Merge

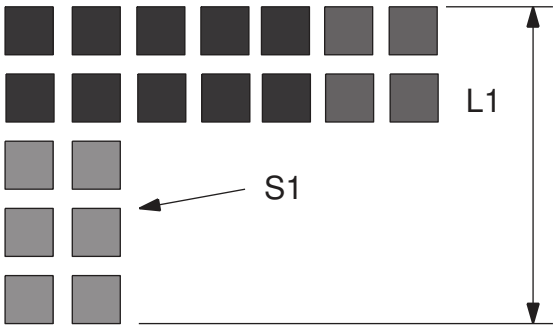


The merged result is shown in Figure 162. The output netlist for this case is as follows:

```
R1 no:1 no:2   (1/10)*via_res    $area=10*via area $lvl= num
R2 no:3 no:4   (1/10)*via_res    $area=10*via area $lvl=num
R3 no:2 no:3   value $w =num $l=num $lvl =num
```

Another possibility is that the tool might divide the vias into three groups as shown in Figure 163.

Figure 163 Case 1 - Dividing Into Three Groups



The output netlist for three groups is as follows:

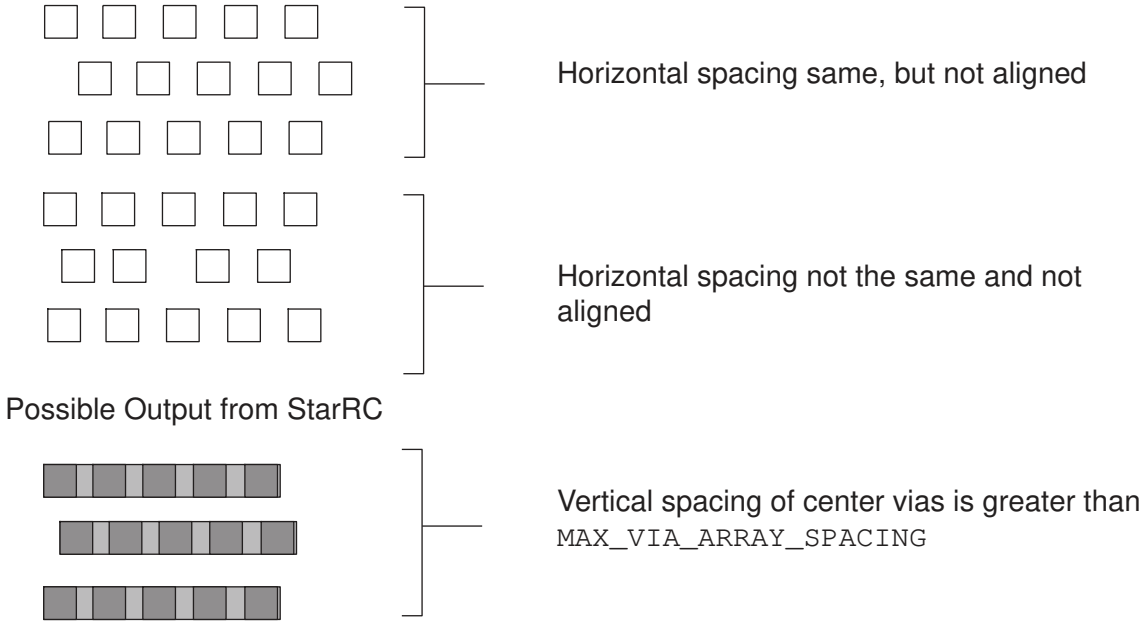
```
R1 no:1 no:2  (1/6)*via_res   $area=6*via_area
R2 no:3 no:4  (1/10)*via_res  $area=10*via_area
R3 no:5 no:6  (1/4)*via_res   $area=4*via_area
R4 no:2 no:3  value $w =num $l=num $lvl =num
R5 no:4 no:5  value $w =num $l=num $lvl =num
```

Feedback

Asymmetric Via Arrays

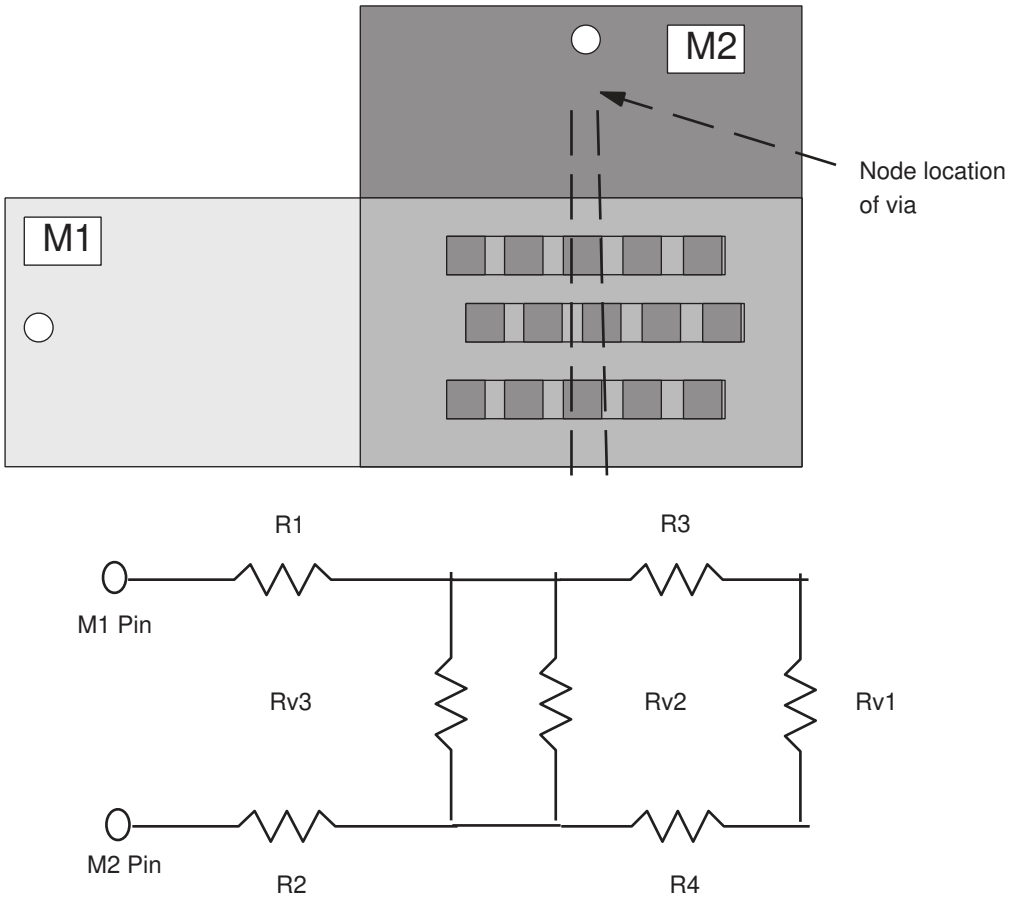
If the design has asymmetric via arrays with different pitch, the tool groups them arbitrarily based on the `MAX_VIA_ARRAY_SPACING` and `MAX_VIA_ARRAY_LENGTH` settings in the mapping file, as shown in Figure 164.

Figure 164 Case 2 - Irregular Horizontal Spacing



Feedback

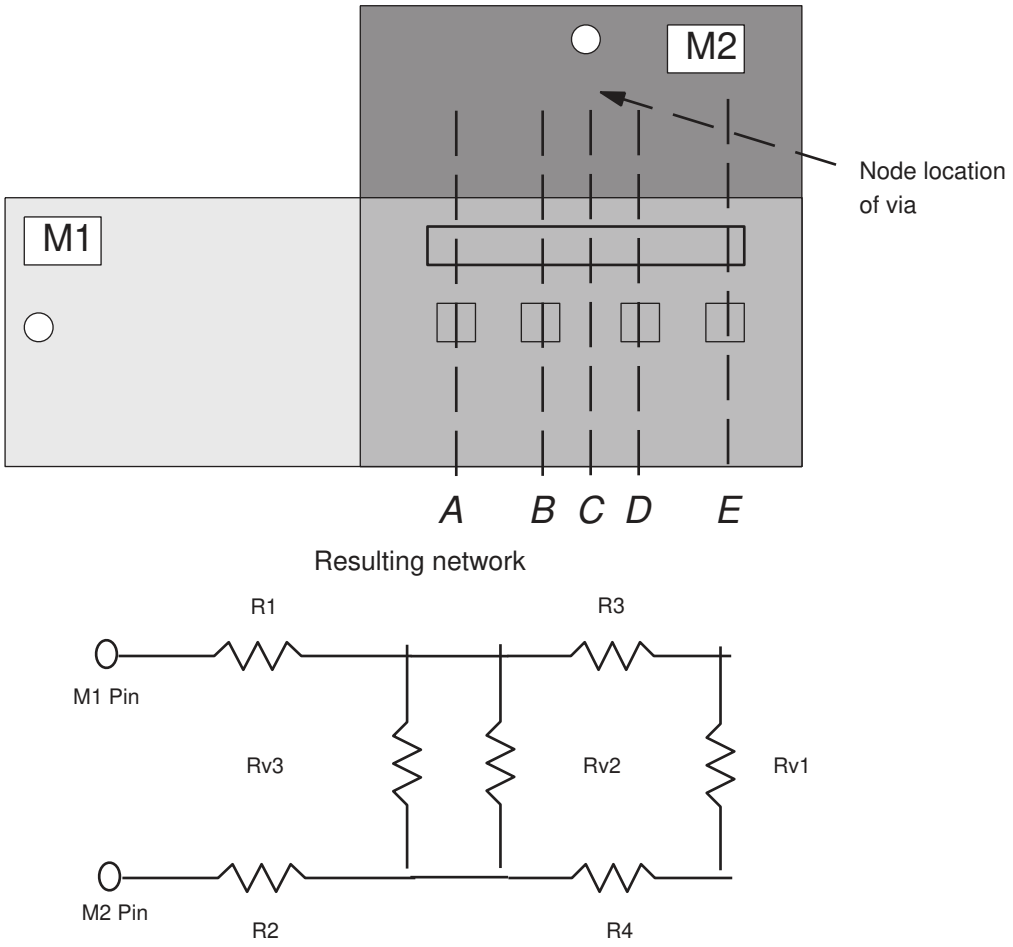
Figure 165 View From Under the Network



In Figure 165, you can possibly get below the network. However, if the distance between two via node locations is small, it can be merged. This means R3 and R4 can be shorted.

Feedback

Figure 166 View From Under the Network - Multiple Nodes



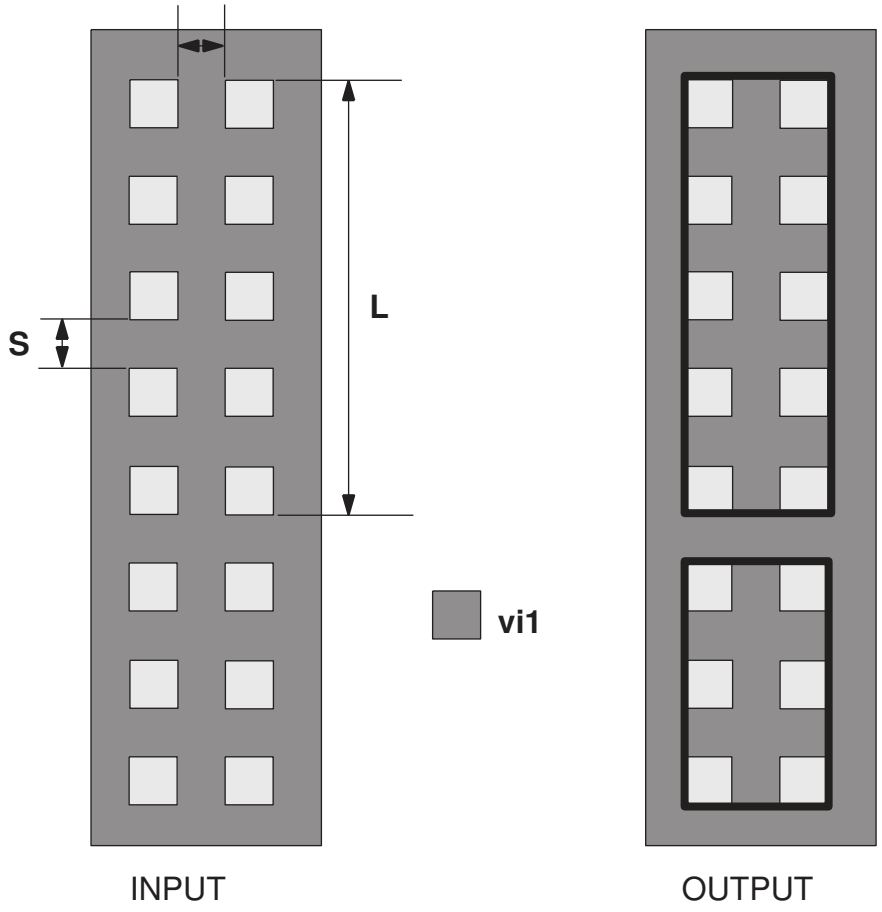
In Figure 166, the StarRC tool creates multiple nodes for the vias. It chooses the center for the via-merged polygon (node C as the merged polygons are aligned). Nodes A, B, D, and E are created for the individual vias. Because the distance between nodes B, C, and D is small, the metal resistance is shorted out and the resulting network is shown in the lower portion of the figure.

Feedback

Rectilinear Via Arrays

In a simple rectilinear via array, as shown in Figure 167, the StarRC tool merges the vias based on the settings of the `MAX_VIA_ARRAY_SPACING` and `MAX_VIA_ARRAY_LENGTH` parameters in the mapping file.

Figure 167 Simple Rectilinear Via Array




```
MAX_VIA_ARRAY_SPACING = S
MAX_VIA_ARRAY_LENGTH = L
```

The output netlist for the arrays shown in Figure 167 is as follows:

```
R1 no:1 no:2    (1/10)*via_res     $area=10*via_area
R2 no:3 no:4    (1/6)* via_res     $area=6* via_area
```

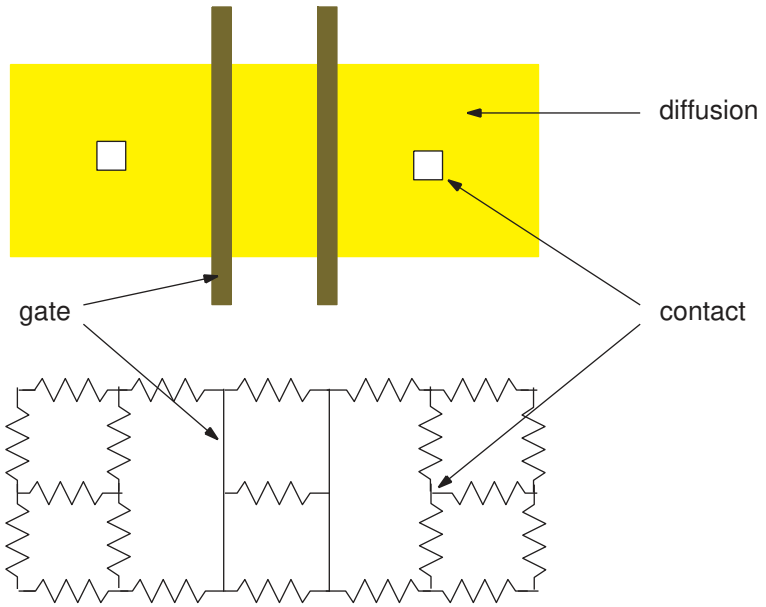
Feedback

Diffusion Resistance

In the ITF file, diffusion is defined as a conductor for a standard shallow trench isolation (STI) process. By default, if diffusion is not defined in the ITF file, no resistance is extracted.

Diffusion resistance is extracted as a mesh by default. The gate and diffusion overlap is located at the equipotential surface (line node), as illustrated in Figure 168.

Figure 168 Resistance Mesh in Source and Drain Diffusion Region (LINE mode)

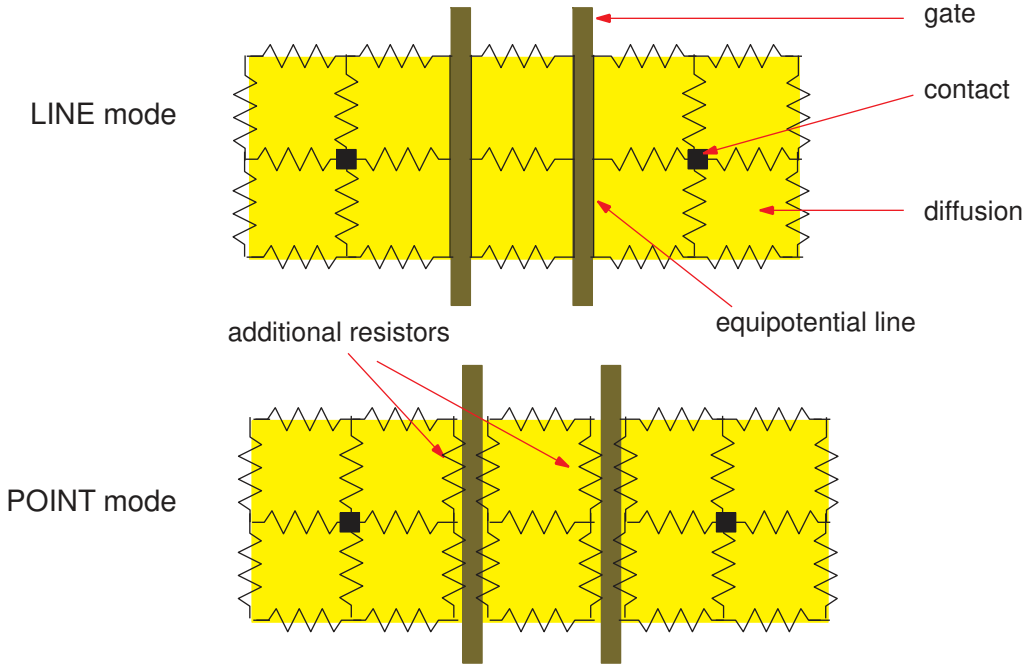


For advanced process nodes and transistors with large widths, the aspect ratio of the diffusion might be very large. You can model the effect with additional resistors along the source and drain contact area by setting the `DIFFUSION_RES_MODE` command to `POINT` (the default is `LINE`).

Feedback

Figure 169 illustrates the two models.

Figure 169 Diffusion Resistance Modes



Feedback

User-Defined Diffusion Resistance

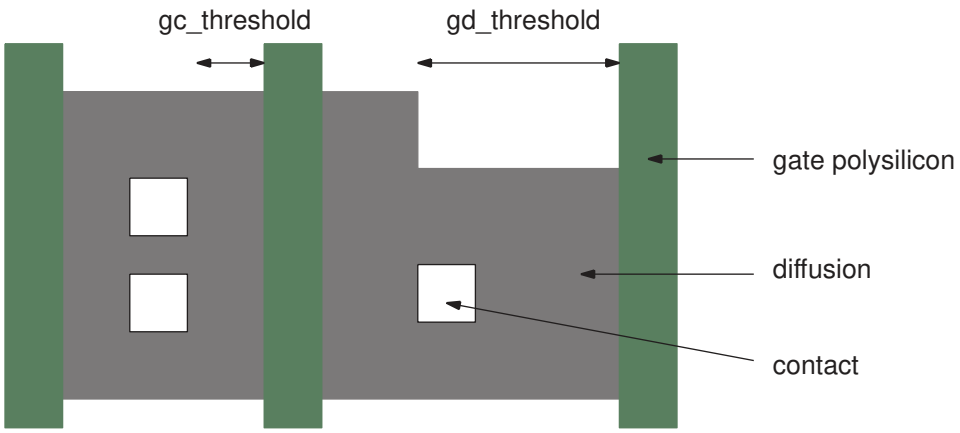
For advanced processing nodes, you can take factors such as contact location and diffusion layout into account by applying a user-defined model to specific diffusion patterns.

To use this analysis, perform the following steps:

1. Set the `USER_DEFINED_DIFFUSION_RESISTANCE` command to `YES` in the StarRC command file. (This step is optional because the command defaults to `YES`.)
2. Include the `USER_DEFINED_DIFFUSION_RESISTANCE` command within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
3. Set the `LAYER_TYPE` keyword to `GATE` within the `CONDUCTOR` block of the gate conductor layer in the ITF file.
4. Specify a model name by using the `diffusion_res_model` keyword in the `conducting_layers` command in the mapping file.

Resistances calculated by the user-defined diffusion resistance method are inserted into the circuit before reduction and are included in reduction operations. To use this method, the layout dimensions must be smaller than the threshold values specified by the `GATE_TO_CONT_THRESHOLD` parameter (labeled gc_threshold in Figure 170) and the `GATE_TO_DIFF_BEND_THRESHOLD` parameter (labeled gd_threshold).

Figure 170 User-Defined Diffusion Resistance Parameters



You can use the `grdgenxo -res_update` command to update the nxtgrd file with new user-defined diffusion resistance parameters.

Feedback

Simultaneous multicorner extraction is supported. However, the `GATE_TO_CONT_THRESHOLD` and `GATE_TO_DIFF_BEND_THRESHOLD` values must be identical between corners. Temperature sensitivity analysis is not supported.

13

ITF Examples

This appendix contains examples of ITF files for several process technologies.

The following topics each contain an ITF file example and a diagram of the process cross section:

- Fully Planar Process ITF Example
- Conformal Dielectric Process ITF Example
- Local Interconnect ITF Example
- Layer Etch Process ITF Example
- Emulated Metal Fill ITF Example
- Transistor-Level Process ITF Example
- Through-Silicon Via ITF Example
- Trench Contact Process ITF Example

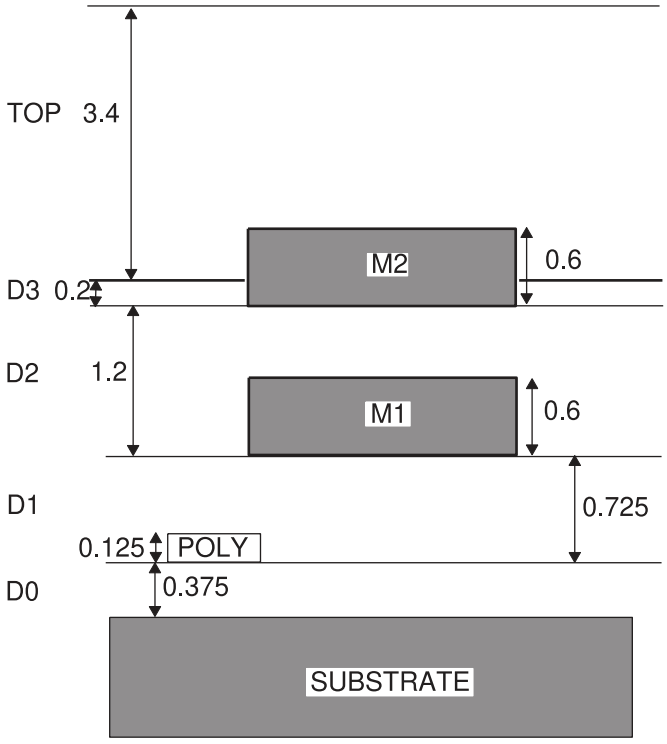
Feedback

Fully Planar Process ITF Example

The following example ITF file describes the fully planar process shown in Figure 171.

```
TECHNOLOGY = planar
DIELECTRIC TOP { THICKNESS=3.4 ER=3.9 }
DIELECTRIC D3 { THICKNESS=0.2 ER=4.7 }
CONDUCTOR M2 { THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05 }
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 { THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05 }
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY{ THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
}
DIELECTRIC D0{ THICKNESS=0.375 ER=3.9 }

VIA sub_tie { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA poly_cont { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA via { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

Figure 171 Fully Planar Process



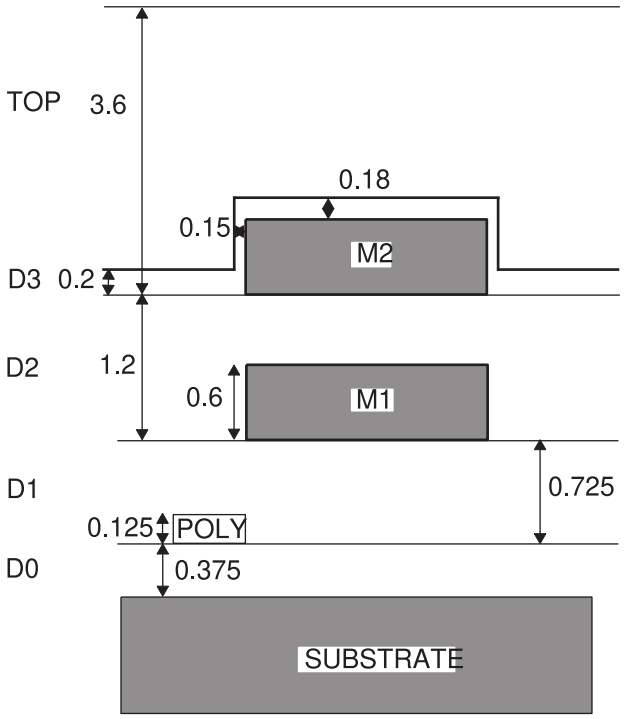
Feedback

Conformal Dielectric Process ITF Example

The following example ITF file describes the conformal dielectric structure shown in Figure 172.

```
TECHNOLOGY = conformal
$ TOP is planarized by measuring from D2
DIELECTRIC TOP { THICKNESS=3.6 MEASURED_FROM=D2 ER=3.9 }
$ D3 is a conformal dielectric
DIELECTRIC D3 {
THICKNESS=0.2 MEASURED_FROM=TOP_OF_CHIP
SW_T=0.15 TW_T=0.18 ER=5.9 }
CONDUCTOR M2 { THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05 }
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 { THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05 }
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY { THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
}
DIELECTRIC D0 { THICKNESS=0.375 ER=3.9 }

VIA DIFF_CONT { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA POLY_CONT { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA V1 { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

Figure 172 Conformal Dielectric Process



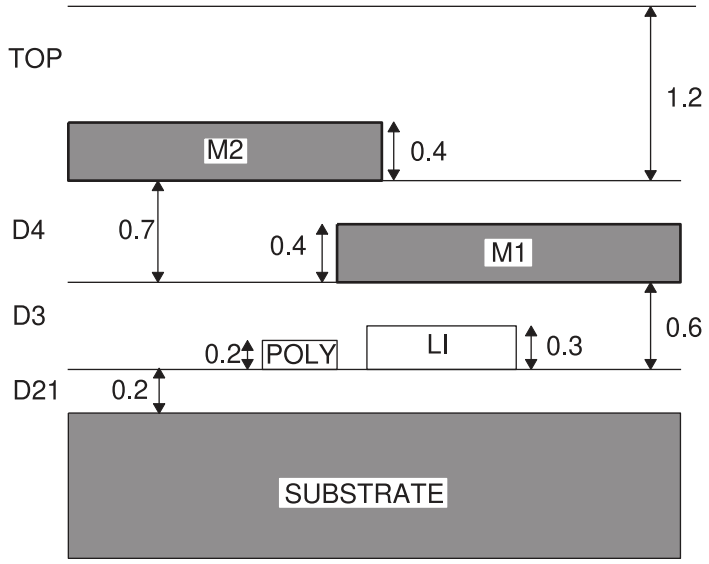
Feedback

Local Interconnect ITF Example

The following example ITF file describes the local interconnect structure shown in Figure 173.

```
TECHNOLOGY = polyli
DIELECTRIC TOP { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M2 { THICKNESS=0.4 WMIN=0.5 SMIN=0.5 RPSQ=0.05
DIELECTRIC D4 { THICKNESS=0.7 ER=3.9 }
$ D4 thickness measured from D3
CONDUCTOR M1 { THICKNESS=0.4 WMIN=0.4 SMIN=0.4 RPSQ=0.05 }
DIELECTRIC D3 { THICKNESS=0.6 ER=3.9 }
$ D3 thickness measured from D21
CONDUCTOR LI { THICKNESS=0.3 WMIN=0.4 SMIN=0.4 RPSQ=1 }
$ LI thickness measured from top of D21
CONDUCTOR POLY { THICKNESS=0.2 WMIN=0.2 SMIN=0.2 RPSQ=10.0 }

$ POLY thickness measured from top of D21

DIELECTRIC D21 { THICKNESS=0.2 ER=3.9 }

VIA LI_SUB { FROM=SUBSTRATE TO=LI AREA=0.25 RPV = 4 }
VIA CONT { FROM=LI TO=M1 AREA=0.25 RPV=5 }
VIA V1 { FROM=M1 TO=M2 AREA=0.25 RPV=4 }
```

Figure 173 Local Interconnect



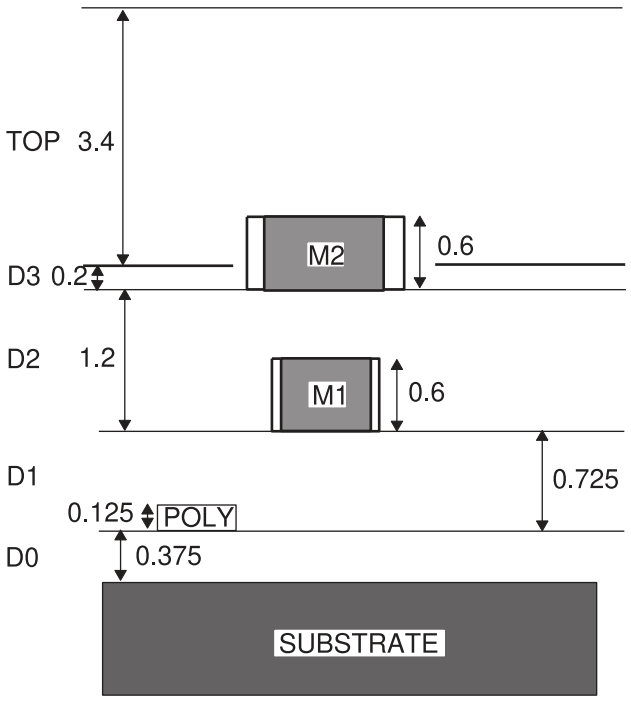
Feedback

Layer Etch Process ITF Example

The following example ITF file describes the layer etch process shown in Figure 174.

```
TECHNOLOGY = etch
DIELECTRIC TOP { THICKNESS=3.4 ER=3.9 }
DIELECTRIC D3 { THICKNESS=0.2 ER=3.9 }
CONDUCTOR M2 {
THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05
ETCH=0.1 }
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 {
THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05
ETCH=0.05 }
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY{ THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
}
DIELECTRIC D0 { THICKNESS=0.375 ER=3.9 }

VIA sub_tie { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA poly_cont { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA via { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

Figure 174 Layer Etch Process



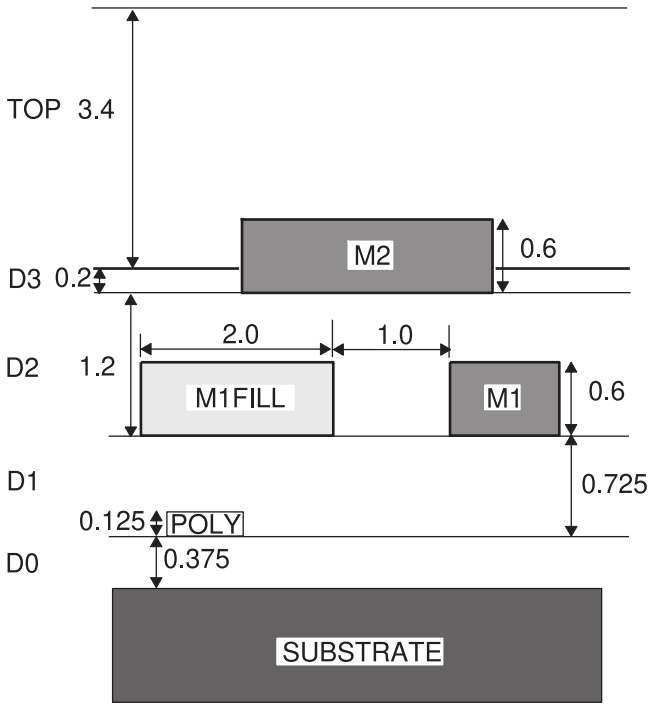
Feedback

Emulated Metal Fill ITF Example

The following example ITF file describes the metal fill structure shown in Figure 175.

```
TECHNOLOGY = metal_fill
DIELECTRIC TOP { THICKNESS=3.4 ER=3.9 }
DIELECTRIC D3 { THICKNESS=0.2 ER=3.9 }
CONDUCTOR M2 {
THICKNESS=0.6 WMIN=0.5 SMIN=0.5 RPSQ=0.05
DIELECTRIC D2 { THICKNESS=1.2 ER=3.9 }
CONDUCTOR M1 {
THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05
FILL_RATIO=0.4 FILL_SPACING=1.0 FILL_WIDTH=2.0 }
DIELECTRIC D1 { THICKNESS=0.725 ER=3.9 }
CONDUCTOR POLY{ THICKNESS=0.125 WMIN=0.3 SMIN=0.3 RPSQ=10.0
}
DIELECTRIC D0 {THICKNESS = 0.375 ER = 5.2}

VIA sub_tie { FROM=SUBSTRATE TO=M1 AREA=0.25 RPV=5 }
VIA poly_cont { FROM=POLY TO=M1 AREA=0.25 RPV=4 }
VIA via { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

Figure 175 Metal Fill Process (Emulated)



Feedback

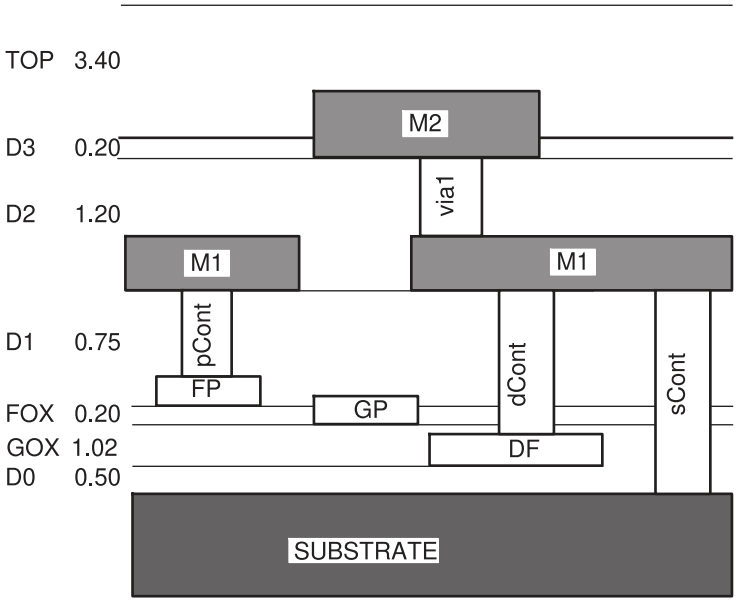
Transistor-Level Process ITF Example

The following example ITF file describes the transistor-level structure shown in Figure 176.

```
TECHNOLOGY=xtor
DIELECTRIC TOP { THICKNESS=3.40 ER=4.3 }
DIELECTRIC D3 { THICKNESS=0.20 ER=3.9 }]
CONDUCTOR M2 { THICKNESS=0.60 WMIN=0.55 SMIN=0.50 RPSQ=0.062
}
DIELECTRIC D2 { THICKNESS=1.20 ER=3.9 }
CONDUCTOR M1 { THICKNESS=0.60 WMIN=0.50 SMIN=0.45 RPSQ=0.062
}

DIELECTRIC D1 { THICKNESS=0.75 ER=3.9 }
CONDUCTOR FP{ THICKNESS=0.30 WMIN=0.35 SMIN=0.40 RPSQ=3.200
}
DIELECTRIC FOX { THICKNESS=0.20 ER=3.9 }
CONDUCTOR GP{ THICKNESS=0.30 WMIN=0.35 SMIN=0.40
RPSQ=3.200 LAYER_TYPE=GATE}
DIELECTRIC GOX { THICKNESS=1.02 ER=5.0 }
CONDUCTOR DF{ THICKNESS=1.00 WMIN=1.00 SMIN=0.35
RPSQ=10.00 }
DIELECTRIC D0 { THICKNESS=0.50 ER=3.9 }

VIA via1 { FROM=M1 TO=M2 RHO=0.263 }
VIA pCont { FROM=FP TO=M1 RHO=0.352 }
VIA dCont { FROM=DF TO=M1 RHO=0.500 }
VIA sCont { FROM=SUBSTRATE TO=M1 RHO=0.600 }
```

Feedback

Figure 176 Transistor-Level Process



Through-Silicon Via ITF Example

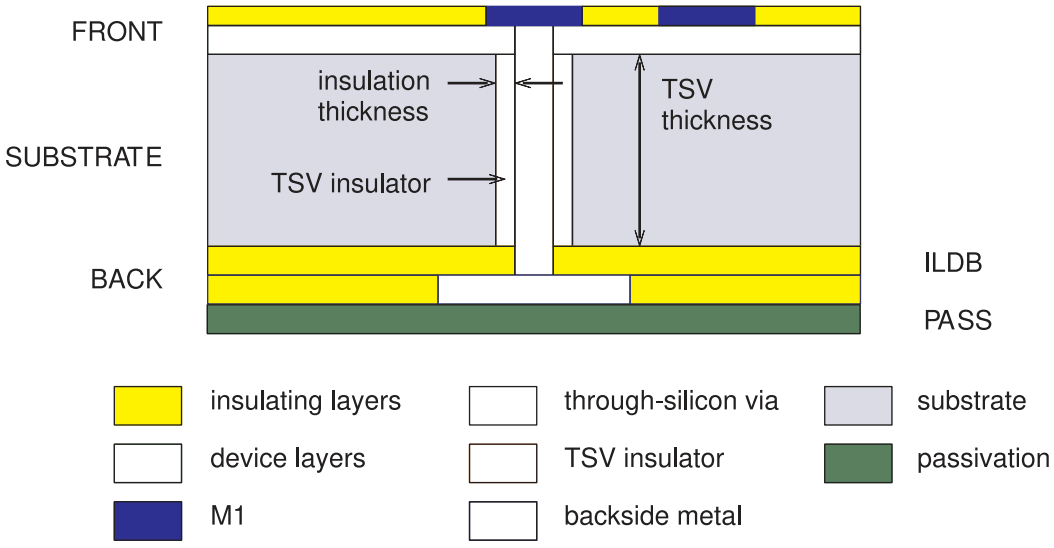
The following example ITF file describes the through-silicon via structure shown in Figure 177. The `TSV` statement must be placed after the frontside ITF statements and before the backside ITF statements.

```
TECHNOLOGY = tsv_process
GLOBAL_TEMPERATURE = 25.0

$ Frontside ITF statements
CONDUCTOR M1 {
  THICKNESS=0.3 WMIN=0.12 SMIN=0.12 SIDE_TANGENT=0.2
  CRT1=7.0e-03 CRT2=-8.0e-07
}
DIELECTRIC ILD_B { ER=5.5 THICKNESS=0.5 }
DIELECTRIC GATE_OXIDE { ER=4.3 THICKNESS=0.03 }
DIELECTRIC FOXIDE_A { ER=5.0 THICKNESS=0.1 }
CONDUCTOR DIFFUSION {
  THICKNESS=0.1 WMIN=0.1 SMIN=0.06
  CRT1=8.0e-03 CRT2=-1.0e-07 RPSQ=36.0
}
DIELECTRIC FOXIDE { ER=5.0 THICKNESS=0.2 }
VIA via1 { FROM=M1 TO=M2 AREA=0.03 RPV=2.5 CRT1=3.0e-04 CRT2=-6.0e-06 }

$ TSV statement
TSV tsv {
```

Feedback

```
  FROM=M1 TO=M1b RHO=0.05 AREA=49.0 THICKNESS=20.0
  INSULATION_THICKNESS = 0.4 INSULATION_ER = 5.5
  CRT1=7.0e-03 CRT2=-3.0e-08
}

$ Backside ITF statements
DIELECTRIC PASS { ER=9.0 THICKNESS=2.0 }
DIELECTRIC DIELb { ER=5.0 THICKNESS=1.0 }
CONDUCTOR M1b {
  THICKNESS=2.0 WMIN=4.0 SMIN=4.0 SIDE_TANGENT=-0.2
  CRT1=1.0e-03 CRT2=-7.0e-07
}
DIELECTRIC ILDB{ ER=5.2 THICKNESS=1.0 }
```

Figure 177 Cross Section of Through-Silicon Via



Trench Contact Process ITF Example

The following example shows a trench contact process definition in the ITF file.

Example 20 Trench Contact Process Definition in the ITF File

```
CONDUCTOR TC {
  THICKNESS=0.05 WMIN=0.025 SMIN=0.060 RPSQ=1.0
  GATE_TO_CONTACT_SMIN=0.02
  LAYER_TYPE=TRENCH_CONTACT
}

DIELECTRIC D4 {THICKNESS=0.015 ER=4.5 MEASURED_FROM=D3 }
```

Feedback

```
DIELECTRIC DC_POLY {
  THICKNESS=0.0 ER=7.0
  IS_CONFORMAL ASSOCIATED_CONDUCTOR=POLY
  SW_T=0.01 TW_T=0.005
}

CONDUCTOR POLY {
  THICKNESS=0.03 WMIN=0.045 SMIN=0.045 RPSQ=1.0
  GATE_TO_CONTACT_SMIN=0.02
  LAYER_TYPE=GATE
}
DIELECTRIC D3 {THICKNESS=0.001 ER=2.8 }
DIELECTRIC D2 {THICKNESS=0.05 ER =3.4 }

CONDUCTOR DIFF {
  THICKNESS=0.05 SMIN=0.045 WMIN=0.06 RPSQ=1.0
  RAISED_DIFFUSION_THICKNESS = 0.015
  RAISED_DIFFUSION_TO_GATE_SMIN = 0.014
  RAISED_DIFFUSION_ETCH = 0.010
  LAYER_TYPE=DIFFUSION
}

DIELECTRIC D1 { THICKNESS=0.1 ER=6.8}
```

Part 3: StarRC Command Reference

14

StarRC Commands

The following reference pages describe the commands that you can use in the StarRC command file.

A line that begins with an asterisk (*) in a command file is a comment. Comment lines do not wrap.

Command names are not case-sensitive. The terms statement, command, and option are synonymous. The term keyword most often refers to a syntax word that is used within a higher-level command or option.

Feedback

3D_IC

Enables the 3-D IC flow.

Syntax

```
3D_IC: YES | NO
```

Arguments

Argument	Description
YES	Enables the 3-D IC flow
NO (default)	Disables the 3-D IC flow

Description

The 3D_IC command enables the 3-D IC flow for modeling through-silicon vias (TSVs).

See Also

- TSV
- 3D_IC_FILTER_DEVICE
- 3D_IC_FLOATING_SUBSTRATE
- 3D_IC_SUBCKT_FILE
- 3D_IC_TSV_COUPLING_EXTRACTION
- TSV_ CELLS
- Through-Silicon Vias

Feedback

3D_IC_FILTER_DEVICE

Specifies a device to be flattened in the netlist during 3-D IC extraction.

Syntax

```
3D_IC_FILTER_DEVICE: device_name
```

Arguments

Argument	Description
`device_name`	Specifies the device to be flattened in the netlist during 3-D IC extraction

Description

The `3D_IC_FILTER_DEVICE` command specifies the device that should be flattened in the netlist during 3-D IC extraction.

For 3-D IC extraction, run layout-versus-schematic (LVS) analysis on each die separately. If you run LVS on several die together, add a pseudo-resistor device at the ports to separate them from the net in interposer and prevent shorts. There is no device in the interposer.

Examples

The following syntax specifies that device ABC_1 should be flattened in the netlist during 3-D IC extraction.

```
3D_IC_FILTER_DEVICE: ABC_1
```

See Also

- TSV
- 3D_IC
- 3D_IC_FILTER_DEVICE_PRIMARY_NETS
- 3D_IC_FLOATING_SUBSTRATE
- 3D_IC_SUBCKT_FILE
- 3D_IC_TSV_COUPLING_EXTRACTION
- TSV_ CELLS
- Through-Silicon Vias

Feedback

3D_IC_FILTER_DEVICE_PRIMARY_NETS

Specifies the name of net to select when the `3D_IC_FILTER_DEVICE_PRIMARY_NETS` command merges two nets together.

Syntax

```
3D_IC_FILTER_DEVICE_PRIMARY_NETS: net_name1 net_name2
```

Arguments

Argument	Description
`net_name1 net_name2`	Specifies the preferred net name to retain Default: none

Description

If two or more nets are reduced by the `3D_IC_FILTER_DEVICE_PRIMARY_NETS` command, the two nets connected to the device are merged. The StarRC tool selects the name of the net as follows:

- Default: Randomly selects one of the net names from the merged nets.
- `3D_IC_FILTER_DEVICE_PRIMARY_NETS`: Selects the primary net name from the list and uses the selected primary net name instead for the merged net.

Examples

The following syntax specifies that the net `ABC_1` should be selected in the netlist during 3-D IC extraction.

```
3D_IC_FILTER_DEVICE_PRIMARY_NETS: ABC_1
```

See Also

- 3D_IC_FILTER_DEVICE

Feedback

3D_IC_FLOATING_SUBSTRATE

Specifies the through-silicon via (TSV) substrate as a floating conductor.

Syntax

```
3D_IC_FLOATING_SUBSTRATE: YES | NO
```

Arguments

Argument	Description
YES	Specifies the TSV substrate as a floating conductor
NO (default)	Specifies that the TSV substrate is not a floating conductor

Description

The 3D_IC_FLOATING_SUBSTRATE command models the TSV substrate in the interposer as a floating conductor. The grdgenxo tool ignores its effect on modeling. You do not need to specify the substrate layer in the ITF file; its thickness is determined by the HEIGHT parameter in a TSV definition.

The StarRC tool uses name substitution for this floating ground. When the 3D_IC_FLOATING_SUBSTRATE command is set to YES, the ground node name is internally set to fsub_starrc and all zero ground nets in the netlist are replaced by fsub_starrc.

This feature is not supported in SPEF netlists because the SPEF does not netlist ground nets explicitly.

See Also

- TSV
- 3D_IC
- 3D_IC_FILTER_DEVICE
- 3D_IC_SUBCKT_FILE
- 3D_IC_TSV_COUPLING_EXTRACTION
- TSV_ CELLS
- Through-Silicon Vias

Feedback

3D_IC_SUBCKT_FILE

Specifies the subcircuits used to represent through-silicon via (TSV) and microbump parasitics.

Syntax

```
3D_IC_SUBCKT_FILE: filename1 filename2 ...
```

Arguments

Argument	Description
`filename1 ...`	The files containing the subcircuits used to represent TSV and microbump extraction results

Description

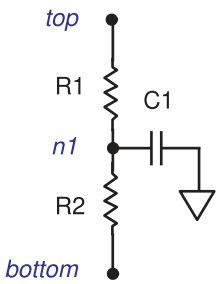
The `3D_IC_SUBCKT_FILE` command specifies one or more files that contain the subcircuits used to represent the internal parasitics of TSVs and microbumps. Each file can contain multiple models.

You can specify models of different types for different TSV layers. If a model is specified multiple times, the StarRC tool takes the definition from the first file read and issues warnings for subsequent definitions of the same model.

R and RC Models for TSV Via and Microbump Replacement

The StarRC tool supports 1R, 2R, and 2R1C models for both via and microbump replacement. When using these models, connect the TSV subcircuit netlist from the front side to the backside metals. Figure 178 shows the layout of the 2R1C model.

Figure 178 2R1C Subcircuit Model



Feedback

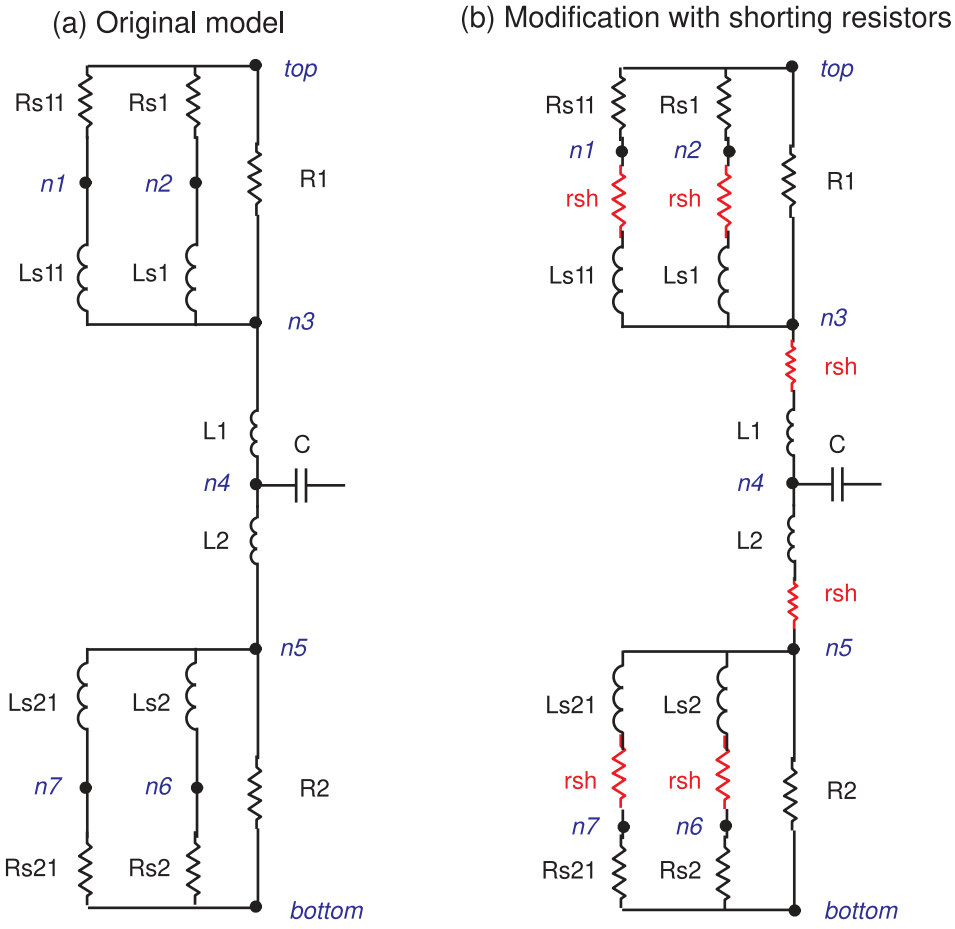
The 2R1C model file is defined as the following, where *name* is a via or cell name:

```
.subckt name top bottom
r1     top n1 0.0126
c1     n1 0 5.678e-3
r2     n1 bottom 0.0126
.ends
```

RLC Model for TSV Via Replacement

For TSV via replacement, the StarRC tool also supports the RLC subcircuit model shown in Figure 179(a). The circuit configuration of the model must be exactly as shown and the element values must be symmetric.

Figure 179 RLC Model for TSV Via Replacement



Feedback

In the RLC model file, you must specify the subcircuit elements in the order shown in the following example:

```
.subckt TSV top bottom
rs11_tsv      top          n1         0.0153
rs1_tsv       top          n2         0.0701
r1_tsv        top          n3         0.1210
ls11_tsv      n1           n3         1.1300e-11
ls1_tsv       n2           n3         4.0300e-12
l1_tsv        n3           n4         1.9450e-11
C_tsv         n4           n8         155e-15
l2_tsv        n4           n5         1.9450e-11
ls2_tsv       n5           n6         4.0300e-12
ls21_tsv      n5           n7         1.1300e-11
r2_tsv        n5           bottom     0.1210
rs2_tsv       n6           bottom     0.0701
rs21_tsv      n7           bottom     0.0153
Rsub          n8           0          1720
Csub          n8           0          3.61e-14
.ends
```

The capacitance C_tsv connects to ground through the parallel combination of Rsub and Csub.

In the output netlist, the StarRC tool represents each inductance element of the original model as a series combination of an inductor and a shorting resistor with a resistance of 0.001 ohms, as illustrated in Figure 179(b).

- If the resistor in the original model has a resistance larger than 0.001 ohms, the model resistance is adjusted by subtracting the value of the shorting resistance to ensure that the total series resistance is the same as specified in the original model.

  In the model example, the value of Rs1 is changed from 0.0701 ohms to 0.0691 ohms when the 0.001-ohm shorting resistor is inserted.

- If the resistor in the original model has a resistance smaller than 0.001 ohms, it is left unchanged. However, the addition of the shorting resistor might alter the intended behavior of the model.

Resistance Variation With Temperature in Through-Silicon Via Layers

To set up temperature coefficients for through-silicon via (TSV) layers, the tool calculates resistances as follows to define temperature-dependent resistance models for TSV layers:

$$R = R0 * \left[1 + TC1 * (T - T0) + TC2 * (T - T0)^2\right]$$

Where,

- R0 is the resistance value at the nominal temperature T0.
- R is the modeled resistance.

Feedback

- T0 is the nominal temperature.
- T is the operating temperature.
- TC1 and TC2 are the parasitic resistor temperature coefficients (linear and quadratic temperature coefficients).

Note the following conditions to set up temperature coefficients for TSV layers:

- You must set `3D_IC` to `YES` and specify the required files with the `3D_IC_SUBCKT_FILE` command. Example 21 lists the order to specify the temperature coefficients:

Example 21 Specifying the temperature coefficients

```
.subckt TSV top bottom
r1            top          n1         0.0126  1.25e-3  -1.95e-7// tc1 tc2
c1            n1           0          0.0056f
r2            n1           bottom     0.0126  1.25e-3  -1.95e-7// tc1 tc2
.ends
```

- The `3D_IC_SUBCKT_FILE` command should include either only resistances or both capacitances and resistances.
- The `OPERATING_TEMPERATURE` command should be set to a temperature that is other than the nominal temperature.
- When all the conditions are met, the StarRC tool evaluates the new R values in place of the original R values that are specified in a 3-D IC .subckt file (specified with the `3D_IC_SUBCKT_FILE` command).

In Example 21,

- The tc1 and tc2 temperature coefficients are set up behind resistances: `1.25e-3 -1.95e-7`.
- When the `operating temperature = 100` and `nominal temperature = 25`, the r1 and r2 resistance values are replaced with `0.01377`. For example, when `derating temperature = 100 - 25 = 75` and `TC1 = 1.25e-3` and `TC2 = -1.95e-7`, the replaced resistance value is `0.01377`.

You can verify RC values for TSV layers in the output netlist files when the tool writes the following information:

- If `operating temperature = nominal temperature`:

```
*|GROUND_NET 0*|
NET netF 0.0254178PF*|
P (netF B 0 199.5000 0.5000)
Cg4_1 netF 0 1.1211e-14
Cg4_2 netF:2 0 1.13715e-14
Cg4_3 netF:3 0 2.37935e-15
Cg4_4 netF:4 0 5.6e-18
```

```
R4_1 netF:3 netF:4 0.0126
R4_2 netF:2 netF:4 0.0126
R4_3 netF netF:3 0.9
R4_4 netF netF:2 2.42856
```

- If `operating temperature = 100` and `nominal temperature = 25`:

```
*|GROUND_NET 0*|
NET netF 0.0254178PF*|
P (netF B 0 199.5000 0.5000)
Cg4_1 netF 0 1.1211e-14
Cg4_2 netF:2 0 1.13715e-14
Cg4_3 netF:3 0 2.37935e-15
Cg4_4 netF:4 0 5.6e-18
R4_1 netF:3 netF:4 0.01377
R4_2 netF:2 netF:4 0.01377
R4_3 netF netF:3 1.12
R4_4 netF netF:2 3.06989
```

Units in Model Files

If the model file does not explicitly state units, the StarRC tool assumes that the units are ohms for resistors, farads for capacitors, and henries for inductors. The tool supports usage of the following scaling factors for units:

- `f` (femto), or 1e-15
- `p` (pico), or 1e-12
- `n` (nano), or 1e-9
- `u` (micro), or 1e-6
- `m` (milli), or 1e-3

Specifying Corners

When you have corner specific `3D_IC_SUBCKT_FILE` command file, note the following points to specify the corners:

- You should specify the file name with absolute path only in the corner file.
- You can specify only one .subckt file for each corner.
- For different corners, you should specify the same number of resistance and capacitance for one through-silicon via (TSV) layer in different .subckt files.

  For example, the StarRC tool issues an error message if you define TSV as 2R1C in one .subckt file and 1R1C in another .subckt file.

To improve extraction with the command and and the results can be quickly evaluated for better designs.You can specify multiple `3D_IC_SUBCKT_FILE` commands in the corner file

Feedback

See Also

- TSV
- 3D_IC
- 3D_IC_FILTER_DEVICE
- 3D_IC_FLOATING_SUBSTRATE
- 3D_IC_TSV_COUPLING_EXTRACTION
- TSV_ CELLS
- Through-Silicon Vias
- CORNERS_FILE
- OPERATING_TEMPERATURE

Feedback

3D_IC_TSV_COUPLING_EXTRACTION

Considers substrate effects during high frequency extractions.

Syntax

```
3D_IC_TSV_COUPLING_EXTRACTION: RCSUB | CEFF | NONE
```

Arguments

Argument	Description
RCSUB	Uses R_{sub} and C_{sub} for SPICE simulation
CEFF	Enables C_{eff} model for static timing analysis simulation
NONE (default)	Does not consider substrate effects during high frequency extractions.

Description

This command specifies the output model when extracting high frequency substrate effects for through-silicon vias.

There are two modes, as follows:

- To use R_{sub} and C_{sub} for SPICE simulation, use the RCSUB setting.

  The StarRC tool evaluates the spacing between two TSV pairs and determines the R_{sub} and C_{sub} values from the ITF table. The netlist is a parallel R_{sub} and C_{sub} network connecting the two TSVs. The netlist is used for SPICE simulation.

- To use C_{eff} for static timing analysis simulation, use the CEFF setting along with the OPERATING_FREQUENCY command.

  The StarRC tool evaluates the spacing between the two TSV pairs and determines the C_{eff} value based on the frequency from the ITF table. The netlist has a single C_{eff} value and connects between two TSVs. You can use this netlist with SPICE and static timing analysis tools.

  The CEFF setting should be used for flows that use the NETLIST_FORMAT: SPEF command.

Examples

The following commands produce a text file for timing analysis.

```
3D_IC_TSV_COUPLING_EXTRACTION: CEFF
3D_IC_SUBCKT_FILE = STA.txt
```

Feedback

See Also

- TSV
- 3D_IC
- 3D_IC_FILTER_DEVICE
- 3D_IC_FLOATING_SUBSTRATE
- 3D_IC_SUBCKT_FILE
- TSV_ CELLS
- Through-Silicon Vias

Feedback

ADD_GATE_ADJUSTMENT_RESISTANCE

Specifies whether to add an adjustment resistance to the gate resistance.

Syntax

```
ADD_GATE_ADJUSTMENT_RESISTANCE: POSITIVE_ONLY | YES | NO
```

Arguments

Argument	Description
`POSITIVE_ONLY` (default)	Inserts only positive adjustment resistance
`YES`	Inserts either positive or negative adjustment resistance
`NO`	Does not insert adjustment resistance

Description

The StarRC tool adds an adjustment resistance to transistor gates if all of the following conditions are true:

- One or more vias land on a gate or gate extension.
- The gate layer is specified in a `GATE_RESISTANCE_ADJUSTMENT_FACTOR` table definition.
- The `ADD_GATE_ADJUSTMENT_RESISTANCE` command is set to `POSITIVE_ONLY` (the default) or `YES` in the StarRC command file.

See Also

- GATE_RESISTANCE_ADJUSTMENT_FACTOR

Feedback

AUTO_RUNSET

Automatically separates device layers based on device definitions in the LVS rule file. Valid only for transistor-level extraction.

Syntax

```
AUTO_RUNSET: NO | YES
```

Arguments

Argument	Description
NO (default)	Processes device layers directly from the LVS database without any detection or modification of layer separation
YES	Enables automatic device layer separation for necessary devices such as MOS devices and capacitors. Generates the corresponding statements for such devices by separating layers based on IGNORE_CAPACITANCE settings.

Description

Normally, the StarRC tool uses device extraction data from LVS tools such as the IC Validator, Hercules, and Calibre tools for device-level parasitic extraction. To ensure accurate extraction results, a rule file for parasitic extraction flows as well as device extraction and LVS flows must abide by the following conventions:

- MOS device and capacitor terminal layers must be distinct from the routing layers that form those terminal layers, and no polygon overlap should occur between those layers. This ensures that the StarRC tool properly excludes all intradevice capacitances that are represented by device models during simulation.
- All contacts must connect exactly two physical layers to ensure uniformity with the physical processes as defined in the StarRC ITF file.

To satisfy these conventions required by the StarRC device-level extraction flow, the LVS rule files must be updated. If you specify the AUTO_RUNSET command, the StarRC tool performs the necessary modifications to your LVS rule file automatically, based on device definitions in the LVS rule file.

Examples

The following syntax shows a typical Calibre rule file that uses the polysilicon interconnect layer directly as the MOS gate terminal layer:

```
gate = poly AND diff
gatenw = gate NOT nwell
```

Feedback

```
ngate = gatenw AND nplus
DEVICE MN(n) ngate poly (G) ndiff (S) ndiff (D) psub (B)
```

In this example, the interconnect layer poly itself serves as the gate terminal layer, while ngate serves as the unconnected recognition (seed) layer. Since the terminal layer derivations violate the tool's assumption that the gate terminal layer represents only the gate area, the StarRC tool erroneously ignores the capacitance for portions of poly that do not serve as part of the gate, that is, all capacitance between the poly interconnect and the device diffusions. The StarRC tool then generates the following instructions for the extraction engine to ignore intradevice capacitance:

```
IGNORE_CAPACITANCE poly ndiff
IGNORE_CAPACITANCE poly psub
```

You can solve this problem by using the `AUTO_RUNSET` command, which performs internal layer separation operations to differentiate the poly serving as the gate terminal from the poly remaining as the routing layer.

Note:

The `AUTO_RUNSET` command can be used only if the recognition layer, which is ngate in this example, exists in the input physical database.

See Also

- IGNORE_CAPACITANCE

Feedback

BLOCK

Defines the layout block name that is to be extracted.

Syntax

```
BLOCK: block_name
```

Arguments

Argument	Description
block_name	The layout name of the block to be extracted Default: none

Description

The usage of the BLOCK option depends on the input design database, as follows:

- NDM format Fusion Compiler or IC Compiler II designs

  The BLOCK option is mandatory. You can optionally specify a label name, if one exists, to identify the specific block to be extracted, in the format block_name/label_name.

- Milkyway designs

  The BLOCK option is mandatory. Valid arguments are top-level blocks or fully placed and routed core blocks that exist in the Milkyway library.

- LEF/DEF designs

  The BLOCK option is invalid. The block to be extracted is determined by the DESIGN keyword in the DEF file specified by the TOP_DEF_FILE command.

- Calibre gate-level and transistor-level flows

  The BLOCK option is mandatory. Valid arguments depend on the settings of the XREF and CELL_TYPE commands. With CELL_TYPE: LAYOUT, which is the default, valid arguments are any cell that exists in the annotated GDSII file and layout netlist file (.nl). With CELL_TYPE:SCHEMATIC and XREF: YES, valid arguments are any valid HCELL from the Calibre compare file (.ixf file).

- Hercules gate-level and transistor-level flows

  The BLOCK option is mandatory. Valid arguments depend on the settings of the XREF and CELL_TYPE commands. With CELL_TYPE:LAYOUT, which is the default, valid arguments are any cell that exists in the WRITE_EXTRACT_VIEW library from Hercules. With CELL_TYPE: SCHEMATIC and XREF:YES or XREF:COMPLETE, valid arguments are any valid equivalence point from Hercules compare.

Feedback

Examples

The following example specifies INT4 as the block to be extracted:

```
BLOCK: INT4
```

The following example specifies the Fusion Compiler or IC Compiler II design MUX2/ver1 as the block to be extracted:

```
BLOCK: MUX2/ver1
```

You can override the `BLOCK` statement for a specific run by using the `-b` option of the `StarXtract` command at the operating system prompt. In the following example, the `-b` option specifies SMALLARRAY as the block to extract instead of the block defined in the command file. The command file is not changed, and the override exists for one run only.

```
% StarXtract -b SMALLARRAY
```

See Also

- CELL_TYPE
- MILKYWAY_DATABASE
- MILKYWAY_EXTRACT_VIEW
- TOP_DEF_FILE
- XREF

Feedback

BLOCK_BOUNDARY

Defines a boundary for the block specified by the BLOCK command.

Syntax

```
BLOCK_BOUNDARY: x1 y1 x2 y2 [… xn yn]
```

Arguments

Argument	Description
x1	The first x-coordinate (in microns)
y1	The first y-coordinate (in microns)
x2	The second x-coordinate (in microns)
y2	The second y-coordinate (in microns)

Description

The BLOCK_BOUNDARY command defines a boundary for the block specified by the BLOCK command when you use the RING_AROUND_THE_BLOCK command for in-context approximation. The BLOCK_BOUNDARY information is used to build the in-context routing approximation rings for preplacement block noise analysis.

The BLOCK_BOUNDARY command is required when the RING_AROUND_THE_BLOCK command is specified for a LEF/DEF flow.

It is not necessary to use the BLOCK_BOUNDARY command for a Milkyway design because the boundary information can be read directly. However, if specified in a Milkyway flow, the BLOCK_BOUNDARY command overrides the design database boundary information.

Specify the coordinates in microns. You can specify an arbitrary number of boundary points. Two points (x1, y1)(x2, y2) define a rectangular block boundary. Specifying more than two points creates a rectilinear (nonrectangular) block boundary. The specified points must be in counterclockwise order. You can specify a closing point, but it is not required. This command can be specified only one time in the StarRC command file.

Examples

The following example shows how to specify a block boundary with four points:

```
BLOCK_BOUNDARY: 0 0 269.6 0 269.6 137.6 0 137.6
```

If the following error message appears, you must provide at least four coordinates of the block boundary, regardless of the shape:

```
ERROR: BLOCK_BOUNDARY must have at least 4 points
```

See Also

- BLOCK
- RING_AROUND_THE_BLOCK
- RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER

Feedback

BUS_BIT

Specifies the character or pair of characters used as the bus bit delimiter during extraction and netlist creation.

Syntax

```
BUS_BIT: {} | [] | () | <> | : | .
```

Arguments

Argument	Description
{}[]()<>:.	Characters used as the bus bit delimiters. Do not insert spaces between the characters in the string . Default: Database bus bit value

Description

The BUS_BIT command specifies the character or pair of characters used as the bus bit delimiter during extraction and netlist creation.

The value of this option affects net selection and the Standard Parasitic Exchange Format (SPEF) *BUS_DELIMITER header statement that is read by follow-on tools. Any literal occurrence of these characters in a net or instance name should be escaped during net selection; attempting to use an illegal delimiter results in an error.

For a LEF/DEF database, the bus bit delimiters are defined by the LEF/DEF default specification or database definition of the BUSBITCHARS keyword in the LEF/DEF database.

This command does not do character substitution in the output netlist; it affects only selection and escaping.

The BUS_BIT command does not define the bus bit character in the final netlist. To make this change in the netlist, you must change the input file or post process the netlist with a script.

See Also

- NETS
- OA_BUS_BIT

Feedback

CALIBRE_LVS_DEVICE_TYPE_CAP

Identifies user-defined capacitor devices for transistor-level extraction using the Calibre LVS flow.

Syntax

```
CALIBRE_LVS_DEVICE_TYPE_CAP: list_of_C_devices
```

Arguments

Argument	Description
list_of_C_devices	List of user-defined CAP devices

Description

This command identifies user-defined capacitors as capacitor devices.

The *list_of_C_devices* argument follows the case-sensitivity set by the CASE_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts.

Examples

```
CALIBRE_LVS_DEVICE_TYPE_CAP: cap_ss
```

See Also

- CALIBRE_LVS_DEVICE_TYPE_MOS
- CALIBRE_LVS_DEVICE_TYPE_RES
- CALIBRE_OPTIONAL_DEVICE_PIN_FILE

Feedback

CALIBRE_LVS_DEVICE_TYPE_MOS

Identifies user-defined MOS devices for transistor-level extraction using the Calibre LVS flow.

Syntax

```
CALIBRE_LVS_DEVICE_TYPE_MOS: list_of_M_devices
```

Arguments

Argument	Description
list_of_M_devices	List of user-defined MOS devices

Description

This command identifies user-defined MOS devices.

The *list_of_M_devices* argument follows the case-sensitivity set by the CASE_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts in certain situations.

Examples

```
CALIBRE_LVS_DEVICE_TYPE_MOS: nmos_ss
```

See Also

- CALIBRE_LVS_DEVICE_TYPE_CAP
- CALIBRE_LVS_DEVICE_TYPE_RES
- CALIBRE_OPTIONAL_DEVICE_PIN_FILE

Feedback

CALIBRE_LVS_DEVICE_TYPE_RES

Identifies user-defined resistors as resistor devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command. Valid only for transistor-level extraction using the Calibre LVS flow.

Syntax

```
CALIBRE_LVS_DEVICE_TYPE_RES: list_of_R_devices
```

Arguments

Argument	Description
list_of_R_devices	List of user-defined RES devices

Description

The `CALIBRE_LVS_DEVICE_TYPE_RES` statement identifies user-defined resistors as resistor devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command.

Examples

In the following example, the rp_sio and pwr_rm1 devices defined in the rule file are identified as resistors:

```
CALIBRE_LVS_DEVICE_TYPE_RES:  rp_sio  pwr_rm1
```

See Also

- CALIBRE_OPTIONAL_DEVICE_PIN_FILE
- WIDE_DEVICE_TERM_RESISTANCE

Feedback

CALIBRE_OPTIONAL_DEVICE_PIN_FILE

Assigns nonstandard device terminals by name for transistor-level extraction using the Calibre LVS flow.

Syntax

```
CALIBRE_OPTIONAL_DEVICE_PIN_FILE: file_name
```

Arguments

Argument	Description
file_name	Name of the device pin file

Description

By default, the StarRC tool assigns nonstandard device terminals by position. However, if you specify the `CALIBRE_OPTIONAL_DEVICE_PIN_FILE` command, the tool assigns the device terminal by the name in the specified file.

The syntax of each line in the file is as follows:

```
device_type model_name[seed_layer] pin_layer_1 (pin name 1) \
     pin_layer_2 (pin name 2) pin_layer_3 (pin name 3) ...
```

Valid values for the `device_type` parameter are M (MOS device), C (capacitor), and R (resistor).

The seed layer must be used when multiple devices have the same model name.

Examples

In the following example, a file named devpin_file contains the list of device terminal names:

```
CALIBRE_OPTIONAL_DEVICE_PIN_FILE: devpin_file
```

Example lines within a device pin file are as follows:

```
M MOS_DEV NDIFSI_D (D) POLY  (G) NDIFSI_S (S) NWELL (B)
C CAP_DEV[met1] CAP_TOP (PLUS) CAP_M1 (MINUS)
C CAP_DEV[met2] CAP_TOP (PLUS) CAP_M2 (MINUS)
```

Feedback

See Also

- CALIBRE_LVS_DEVICE_TYPE_CAP
- CALIBRE_LVS_DEVICE_TYPE_MOS
- CALIBRE_LVS_DEVICE_TYPE_RES

Feedback

CALIBRE_PDBA_FILE

Specifies the use of data from a Calibre Push Down Back-Annotation (PDBA) file. Valid only for transistor-level extraction.

Syntax

```
CALIBRE_PDBA_FILE: file_name
```

Arguments

Argument	Description
file_name	File containing devices and device properties Default: none

Description

The CALIBRE_PDBA_FILE command reads a Calibre PDBA file. This file is generated by the Calibre tool; it lists devices and device properties. Use the CALIBRE_PDBA_FILE command to retrieve the separated properties for a final parasitic netlist with complete device information.

The CALIBRE_PDBA_FILE command might not be necessary, depending on the tool versions and the flow.

- If you are using the StarRC-CCI Push Down Back-annotation (PDBA) flow with StarRC version 2013.12-1 (or later) and Calibre version 2013.4-15.12 (or later), StarRC can obtain the PDBA information from the query file and the LVS extraction report file. To use this capability, include the following command in the Calibre query file:

  ```
  LVS SETTINGS REPORT WRITE cci_file
  ```

  Include the following commands in the Calibre runset:

  ```
  LVS PUSH DEVICES YES
  LVS PUSH DEVICES SEPARATE PROPERTIES "pdba.data"" AGF
  ```

  You do not need to include the CALIBRE_PDBA_FILE command in the StarRC command file because the query file finds it automatically. If the command file includes a CALIBRE_PDBA_FILE command, the file specified by the command takes precedence over the PDBA file from the query output.

- If you are using the StarRC-CCI Push Down Back-annotation (PDBA) flow with earlier versions of either the StarRC or Calibre tools, include the CALIBRE_PDBA_FILE

command in the StarRC command file and the following commands in the Calibre runset:

```
LVS PUSH DEVICES YES
LVS PUSH DEVICES SEPARATE PROPERTIES "pdba.data" AGF
```

- If you are using the StarRC-CCI Push Down Separated Properties (PDSP) flow, do not include the CALIBRE_PDBA_FILE command in the StarRC command file and include the following commands in the Calibre runset:

  ```
  LVS PUSH DEVICES YES
  LVS PUSH DEVICES SEPARATE PROPERTIES YES
  ```

  The StarRC tool automatically uses the PDSP file from the Calibre query output. If the StarRC command file includes a CALIBRE_PDBA_FILE command, the file specified by the command takes precedence over the PDSP file from the query output.

If the CALIBRE_PDBA_FILE command is used with the NETLIST_IDEAL_SPICE_HIER: YES command, the CALIBRE_PDBA_FILE command is ignored and the layout netlist hierarchy is preserved.

When you use the CALIBRE_PDBA_FILE command, the SKIP_CELLS command might fail because the DFM properties annotation can promote the instance hierarchy.

Examples

```
CALIBRE_PDBA_FILE: ./pdba.data
```

See Also

- SKIP_CELLS
- LVS_EXTRACTION_REPORT_FILE
- NETLIST_IDEAL_SPICE_FILE

Feedback

CALIBRE_QUERY_FILE

Specifies the location of Calibre Connectivity Interface input files. Valid only for transistor-level extraction.

Syntax

```
CALIBRE_QUERY_FILE: query_script_file
```

Arguments

Argument	Description
query_script_file	The command file used with the Calibre Connectivity Interface server Default: none

Description

To have the Calibre Connectivity Interface generate all the files needed for StarRC extraction, all the necessary query commands must be in the query command file specified by the `CALIBRE_QUERY_FILE` command.

Examples

Use the command as shown here. An example of a Calibre query file follows.

```
CALIBRE_QUERY_FILE: query.cmd
```

Note:

The `LNXF` construct of the `NET XREF WRITE` command and the `EXPAND_CELLS` construct of the `LAYOUT NAMETABLE WRITE` command require Calibre version 2014.3 or later.

The following example shows a Calibre query command file.

Example 22 Calibre Query File

```
# Define property numbers in annotated GDSII
GDS NETPROP NUMBER 5
GDS PLACEPROP NUMBER 6
GDS DEVPROP NUMBER 7

# Output seed polygon with net ID
GDS SEED PROPERTY ORIGINAL

# Output annotated GDSII mapping file for StarRC
RESPONSE FILE GDS_MAP
GDS MAP
```

```
RESPONSE DIRECT

# Output annotated GDSII file for StarRC
GDS WRITE agf

# Output device table file containing device layer descriptions
RESPONSE FILE devtab_file
DEVICE TABLE
RESPONSE DIRECT

# Output layout net name and net ID mapping table for StarRC
# The EXPAND_CELLS keyword for LAYOUT NAMETABLE WRITE ensures that the
# lnn file and the netlist have the same hierarchy
LAYOUT NETLIST TRIVIAL PINS YES
LAYOUT NETLIST EMPTY CELLS YES
LAYOUT NETLIST NAMES NONE
LAYOUT NAMETABLE WRITE lnn_file EXPAND_CELLS

# Output ideal layout netlist for StarRC
# AGF is the only allowed keyword for LAYOUT NETLIST HIERARCHY
LAYOUT NETLIST PRIMITIVE DEVICE SUBCKTS NO
LAYOUT NETLIST PIN LOCATIONS YES
LAYOUT NETLIST HIERARCHY AGF
LAYOUT NETLIST WRITE netlist_file

# Output net or device instance cross referencing tables for StarRC
NET XREF WRITE nxf_file LNXF
INSTANCE XREF WRITE ixf_file

# Output ports file for StarRC
PORT TABLE CELLS WRITE ports_cells_file

# Output cell extents file for StarRC
CELL EXTENTS WRITE extents.txt
```

See Also

- CALIBRE_RUNSET
- The Calibre Connectivity Interface Flow
- Running the Calibre Query Server

Feedback

CALIBRE_RUNSET

Specifies the LVS command file used for transistor-level extraction using a Calibre LVS flow.

Syntax

```
CALIBRE_RUNSET: lvs_command_file
```

Arguments

Argument	Description
lvs_command_file	The LVS command file used for the Calibre run Default: none

Description

The CALIBRE_RUNSET command specifies the LVS command file used for a Calibre run.

An LVS rule deck contains data creation commands, device creation commands, device property calculations, layer connect sequences, and LVS comparison options. The StarRC tool parses the layer connect sequence from the Calibre runset, including derived layer connectivity, to understand the runset layer connectivity.

The CALIBRE_RUNSET command usage requirements are as follows:

- The command is required if you are using StarRC versions earlier than 2013.12-1.
- The command is required if you are using Calibre versions earlier than 2013.4-15.12.
- If you are using StarRC version 2013.12-1 (or later) and Calibre version 2013.4-15.12 (or later), the StarRC CALIBRE_QUERY_FILE command obtains the necessary information from the query file and the LVS extraction report file. The Calibre query file must also use the LVS SETTINGS REPORT WRITE command to write an extraction report.

  In this case, the StarRC tool ignores the CALIBRE_RUNSET and CALIBRE_PDBA_FILE commands.

You can read a specific Calibre extraction report instead of the automatically determined report by using the StarRC LVS_EXTRACTION_REPORT_FILE command.

Feedback

See Also

- LVS_EXTRACTION_REPORT_FILE
- CALIBRE_QUERY_FILE
- MAPPING_FILE

Feedback

CAPACITOR_TAIL_COMMENTS

Writes geometric information about parasitic capacitors as comments in the netlist. Valid in transistor-level flows only.

Syntax

```
CAPACITOR_TAIL_COMMENTS: YES | NO
```

Arguments

Argument	Description
YES	Writes capacitor geometric information as comments in the netlist
NO (default)	Disables capacitor tail comments

Description

The CAPACITOR_TAIL_COMMENTS command controls whether geometric information about parasitic capacitors is added to the netlist output. This option is available for the SPF, NETNAME, and STAR netlist formats.

The additional information is as follows:

- Layer numbers (one layer for grounded capacitors, two layers for coupling capacitors)

  The $lvl argument is a number that appears in the LAYER_MAP section of the netlist, just after the header. The number is associated with a retained mapping file layer name.

- Instance pin declarations

  For RC extraction, instance pin declarations always appear in the netlist. For capacitance-only extraction, instance pin declarations are not included in the netlist by default. However, if you set the CAPACITOR_TAIL_COMMENTS command to YES, the StarRC tool performs RC extraction regardless of the setting of the EXTRACTION command. This is necessary to obtain instance pin information for the capacitors.

If you specify capacitance-only extraction, resistor information is not included in the netlist even if it is calculated in the background as a result of enabling the capacitor tail comments feature. In addition, the runtime and star directory size for capacitance-only extraction are not smaller than the corresponding values for RC extraction.

You can set the CAPACITOR_TAIL_COMMENTS and NETLIST_TAIL_COMMENTS commands independently. However, the two commands require different restrictions on the REDUCTION command.

Feedback

The following restrictions apply:

- For capacitance-only extraction, the REDUCTION command is automatically set to LAYER regardless of any settings in the command file.
- For RC extraction, the allowed REDUCTION command settings are LAYER, NO, and LAYER_NO_EXTRA_LOOPS.
- The EXTRA_GEOMETRY_INFO command is automatically set to include the NODE option. If the command was previously set to RES, the new setting is NODE RES.
- When this command is used with the field solver (with the FS_EXTRACT_NETS command), there is a small difference between the field solver extracted capacitance and the capacitance from the standard StarRC flow.

Examples

The following example shows an SPF netlist for RC extraction with the capacitor tail comments feature enabled, highlighting the information that would not otherwise appear:

```
*|NETZCB 0.000282157PF
*|I(F82 M12 SRC B 0 0.9 1.63) // $llx=0.9 $lly=1.485 $urx=1.12 $ury=1.775
 $lvl=37
*|I(F58 M6 SRC B 0 0.9 0.55) // $llx=0.9 $lly=0.55 $urx-1.12 $ury=0.55
 $lvl=32
*|S(213 1.025 0.5075) // $llx-0.975 $lly=0.465 $urx=1.075 $ury=0.55
 $lvl=89
C29_69 F82 444 1e-18 $lvl1=37 $lvl2=59
C29_70 F58 444 1e-18 $lvl1=32 $lvl2=59
Cg29_78 F58 0 1e-17 $lvl=32
R29_97 F82 213 10 $l=0.485 $2=0.1 $lvl=89
R29_102 213 F58 20 $a=0.0081 $lvl=171
```

The following example shows the same netlist for capacitance-only extraction with the capacitor tail comments feature enabled, highlighting the information that would not otherwise appear:

```
*|NETZCB 0.000282157PF
*|I(F82 M12 SRC B 0 0.9 1.63) // $llx=0.9 $lly=1.485 $urx=1.12 $ury=1.775 $lvl
=37
*|I(F58 M6 SRC B 0 0.9 0.55) // $llx=0.9 $lly=0.55 $urx-1.12 $ury=0.55 $lvl=32
*|S(213 1.025 0.5075) // $llx-0.975 $lly=0.465 $urx=1.075 $ury=0.55 $lvl=89
C29_69 F82 444 1e-18 $lvl1=37 $lvl2=59
C29_70 F58 444 1e-18 $lvl1=32 $lvl2=59
Cg29_78 F58 0 1e-17 $lvl=32
```

See Also

- NETLIST_FILE
- NETLIST_FORMAT

Feedback

- EXTRA_GEOMETRY_INFO
- REDUCTION
- NETLIST_TAIL_COMMENTS

Feedback

CASE_SENSITIVE

Specifies the case-sensitivity of net and cell names during selection.

Syntax

```
CASE_SENSITIVE: YES | NO
```

Arguments

Argument	Description
YES (default)	Specifies that cell and net names are case-sensitive during selection
NO	Specifies that cell and net names are not case-sensitive

Description

The CASE_SENSITIVE command specifies whether net and cell names are case-sensitive during selection. The case sensitivity of the input database is always retained for netlist creation.

If IGNORE_CASE is set to TRUE in your Hercules runset, then you must specify CASE_SENSITIVE: NO in the StarRC command file.

Examples

The following syntax specifies that all net selection and cell selection are not case-sensitive.

```
CASE_SENSITIVE: NO
```

See Also

- CONLY_NETS
- INSTANCE_PORT
- NETLIST_SELECT_NETS
- NETLIST_TYPE
- NETS
- POWER_NETS
- SKIP_CELLS

Feedback

CELL_TYPE

Specifies whether layout or schematic cell names are used for data selection.

Syntax

```
CELL_TYPE: LAYOUT | SCHEMATIC
```

Arguments

Argument	Description
`LAYOUT` (default)	Uses layout cell names
`SCHEMATIC`	Uses schematic cell names matched during LVS processing

Description

The `CELL_TYPE` command specifies whether layout or schematic cell names are used for blocks and skip cells during data selection.

This command is ignored if `XREF: NO` is specified.

Note:

The `CELL_TYPE` command identifies only the source of cell names for block and skip cell selection. It does not affect the reported cell names.

Examples

```
CELL_TYPE: LAYOUT
```

See Also

- BLOCK
- MILKYWAY_EXTRACT_VIEW
- NET_TYPE
- SKIP_CELLS
- XREF

Feedback

CELLS_IN_SHORTREPORT

Provides information of blockage and skip cells in the shorts summary (shorts_all.sum) file.

Syntax

```
CELLS_IN_SHORTREPORT: YES | NO
```

Arguments

Argument	Description
`Yes` (default)	Cell and instantiated cell names are included in the shorts summary file
`No`	Cell and instantiated cell names are not included in the shorts summary file

Description

The shorts summary file (shorts_all.sum) contains information about blockages and skip cells when the `CELLS_IN_SHORTREPORT` command is set to `YES`.

You can disable the additional information by setting the `CELLS_IN_SHORTREPORT` command to `NO`. When CELLS_IN_SHORTREPORT is set to NO, a short between a signal net and a blockage or skip cell is reported as follows:

```
Short between net <name1> and <name2> Layer = <layer> BBox=(<coords>)
```

The tool also creates another file named shorts_by_cell.sum. The file contains all the instances reported in the shorts_all.sum file. Each line corresponds to one instantiated cell and reports the cell name, the number of shorts within the cell, and the names of the shorted nets within the cell. The nets are sorted by the number of shorts, in decreasing order. Reports only the top ten nets by the number of shorts. The format is as follows:

```
Instantiated_cell Nb_shorts Top_ten_nets
<cell_name> <no_of_shorts> <net0> <net1> ... <net10>
```

For example,

```
Instantiated_cell Nb_shorts Top_ten_nets
cell0 6 cell0/n1 cell0/n2 cell0/n12 cell0/n14 cell0/n20 cell0/n21
```

Feedback

Examples

Shorts of a signal net with the blockage or skip cell in the *shorts_all.sum* file are reported in the following format:

```
Short between net xyz and net yzy_295 of instance <top/ceLL1> of cell
 macro2 Layer=M6 Bbox=(444.204,427.342),
(444.204,427.387)
```

The coordinates for the bounding boxes are in microns. Coordinates might be scaled or unscaled depending on the settings of the `HALF_NODE_SCALE_FACTOR`, `MAGNIFICATION_FACTOR`, and `NETLIST_UNSCALED_COORDINATES` commands.

See Also

- Shorts Reports
- ENHANCED_SHORT_REPORTING
- HALF_NODE_SCALE_FACTOR
- MAGNIFICATION_FACTOR
- NETLIST_UNSCALED_COORDINATES

Feedback

CHECK_SKIP_PCELL_LAYER_NAMES

Enables checking the validity of PCell layer names in a Calibre Connectivity Interface flow.

Syntax

```
CHECK_SKIP_PCELL_LAYER_NAMES: YES | NO
```

Arguments

Argument	Description
YES	Checks layer name validity
NO (default)	Disables the layer name check

Description

The CHECK_SKIP_PCELL_LAYER_NAMES command enables existence checking for blocking layers listed in the file specified by the SKIP_PCELL_LAYERS_FILE command. If the CHECK_SKIP_PCELL_LAYER_NAMES command is set to YES, the StarRC tool checks to see if the blocking layer names are defined in the layout-versus-schematic (LVS) tool database.

Errors

The StarRC tool issues a warning message if all of the following conditions are true:

- The CHECK_SKIP_PCELL_LAYER_NAMES command is set to YES.
- A layer name appears in the BLOCKING_LAYERS section of the file specified by the SKIP_PCELL_LAYERS_FILE command.
- The layer name does not appear in the LVS database file.

See Also

- SKIP_PCELLS
- SKIP_PCELL_LAYERS_FILE

Feedback

CLOCK_NET_ADVANCED_MODEL

Enables advanced clock net inductance modeling features.

Syntax

```
CLOCK_NET_ADVANCED_MODEL: YES | NO
```

Arguments

Argument	Description
YES	Enables advanced modeling features for clock net inductance modeling
NO (default)	Disables advanced clock net inductance modeling features

Description

Setting the CLOCK_NET_ADVANCED_MODEL command to YES enables optional features of clock net extraction . The CLOCK_NET_INDUCTANCE command must also be set to YES to use these features. The analysis requirements for standard inductance extraction also apply to advanced inductance extraction.

The advanced features are as follows:

- Include frequency-dependent effects by using the CLOCK_NET_FREQUENCY and HIGH_CLOCK_NET_FREQUENCY commands.
- Model return current that occurs in shield conductors beyond the immediately adjacent shield conductors by using the CLOCK_NET_SHIELD_EXT_FACTOR command.
- Specify a layer to use as a ground layer for return current by using the CLOCK_NET_INDUCTANCE_GND_LAYER command.

See Also

- CLOCK_NET_INDUCTANCE
- CLOCK_NET_FREQUENCY
- HIGH_CLOCK_NET_FREQUENCY
- CLOCK_NET_SHIELD_EXT_FACTOR
- CLOCK_NET_INDUCTANCE_GND_LAYER
- Clock Net Inductance Extraction

Feedback

CLOCK_NET_FREQUENCY

Specifies the analysis frequency for clock net inductance extraction.

Syntax

```
CLOCK_NET_FREQUENCY: freq
```

Arguments

Argument	Description
freq	Clock frequency, in GHz Default: 1 Allowable range: 0.1 to 20

Description

The CLOCK_NET_FREQUENCY command specifies the frequency used for clock net inductance analysis. This command has an effect only if the CLOCK_NET_INDUCTANCE command is set to YES.

The frequency specified by this command is taken into account as follows:

- If the CLOCK_NET_ADVANCED_MODEL command is set to NO (the default), the StarRC tool calculates one resistance value at zero frequency and one inductance value at the frequency specified by the CLOCK_NET_FREQUENCY command for each net segment.
- If the CLOCK_NET_ADVANCED_MODEL command is set to YES and the HIGH_CLOCK_NET_FREQUENCY command is not set, the tool calculates one resistance value and one inductance value for each net segment at the frequency specified by the CLOCK_NET_FREQUENCY command.
- If the CLOCK_NET_ADVANCED_MODEL command is set to YES and the HIGH_CLOCK_NET_FREQUENCY command is set to a value larger than the CLOCK_NET_FREQUENCY command value, the tool calculates frequency-dependent resistance and inductance values for each net segment. The result is a netlist that can be used for frequency analysis in downstream simulation tools. The model is valid between the two specified frequencies.

See Also

- HIGH_CLOCK_NET_FREQUENCY
- CLOCK_NET_ADVANCED_MODEL
- Clock Net Inductance Extraction

Feedback

CLOCK_NET_INDUCTANCE_GND_LAYER

Specifies a design database layer to serve as a ground plane for clock net inductance analysis.

Syntax

```
CLOCK_NET_INDUCTANCE_GND_LAYER: db_layer
```

Arguments

Argument	Description
db_layer	Conductor layer to use as a ground plane. Not case-sensitive; no wildcards allowed. Default: none

Description

If the CLOCK_NET_INDUCTANCE_GND_LAYER command is used, the specified conductor layer serves as a ground plane during clock net inductance analysis.

To use this factor, both the CLOCK_NET_INDUCTANCE and CLOCK_NET_ADVANCED_MODEL commands must be set to YES.

See Also

- CLOCK_NET_INDUCTANCE
- CLOCK_NET_ADVANCED_MODEL
- CLOCK_NET_SHIELD_EXT_FACTOR
- Clock Net Inductance Extraction

Feedback

CLOCK_NET_INDUCTANCE

Enables clock net inductance extraction.

Syntax

```
CLOCK_NET_INDUCTANCE: YES | NO
```

Arguments

Argument	Description
YES	Enables clock net inductance analysis
NO (default)	Disables clock net inductance analysis

Description

Set the CLOCK_NET_INDUCTANCE command to YES to enable clock net inductance extraction. Inductance analysis is performed in targeted runs separate from standard extraction runs.

In typical usage, you use the NETS command to specify the clock nets of interest. You can optionally specify the frequency by using the CLOCK_NET_FREQUENCY command and the database layers to consider by using the CLOCK_NET_INDUCTANCE_LAYERS command. A clock net and its return path must reside on the same database layer, but different clock nets can reside on different layers.

Clock nets are assumed to be fully shielded by power or ground nets and the current return path is exclusively through the shielding nets. If the return path is not found or is more than 5 microns away, the distance to the shield is modeled to be 5 microns.

Limitations are as follows:

- This feature is available only for gate-level Milkyway, LEF/DEF, Fusion Compiler, or IC Compiler II flow.
- Simultaneous multicorner extraction and power net extraction must be disabled.
- Only DSPF netlists are supported for output.

See Also

- CLOCK_NET_ADVANCED_MODEL
- CLOCK_NET_FREQUENCY

Feedback

- CLOCK_NET_INDUCTANCE_LAYERS
- Clock Net Inductance Extraction

Feedback

CLOCK_NET_INDUCTANCE_LAYERS

Specifies the design database layers to consider for the inductance return paths in clock net inductance extraction.

Syntax

```
CLOCK_NET_INDUCTANCE_LAYERS: layer_list
```

Arguments

Argument	Description
layer_list	Space-delimited list of design database layers to consider for the return paths. Not case-sensitive; no wildcards allowed. Default: all layers

Description

If the CLOCK_NET_INDUCTANCE_LAYERS command is used, only the specified design database layers are used for clock net inductance extraction. A clock net and its return path must reside on the same database layer, but different clock nets can reside on different layers.

To use this command, the CLOCK_NET_INDUCTANCE command must be set to YES.

See Also

- CLOCK_NET_FREQUENCY
- CLOCK_NET_INDUCTANCE
- CLOCK_NET_ADVANCED_MODEL
- Clock Net Inductance Extraction

Feedback

CLOCK_NET_SHIELD_EXT_FACTOR

Specifies a factor to represent the relative conductivity of secondary return paths for clock net inductance analysis.

Syntax

```
CLOCK_NET_SHIELD_EXT_FACTOR: cond_factor
```

Arguments

Argument	Description
cond_factor	Relative conductivity factor top apply to external shields. Default: 0.5 Range: 0.0 to 1.0

Description

If the CLOCK_NET_SHIELD_EXT_FACTOR command is used, the specified factor represents the effective conductivity of metal return paths beyond the nearest neighbor shields.

To use this factor, both the CLOCK_NET_INDUCTANCE and CLOCK_NET_ADVANCED_MODEL commands must be set to YES.

See Also

- CLOCK_NET_INDUCTANCE
- CLOCK_NET_ADVANCED_MODEL
- CLOCK_NET_INDUCTANCE_GND_LAYER
- Clock Net Inductance Extraction

Feedback

COMPARE_DIRECTORY

Specifies the location of the Hercules LVS COMPARE output.

Syntax

```
COMPARE_DIRECTORY: path
```

Arguments

Argument	Description
path	The path to the location of the Hercules LVS COMPARE output files Default: none

Description

The COMPARE_DIRECTORY command specifies the location of the Hercules LVS COMPARE output. This path is specified in the Hercules runset HEADER section, with the COMPARE_DIRECTORY option. The Hercules default for this option is ./run_details/compare.

This command is optional; however, if this path is not specified and XREF: YES or XREF: COMPLETE is specified, the tool attempts to read the compare directory location from the XTR (Milkyway Extract) view.

See Also

- XREF

Feedback

CONLY_NETS

Reports cross-capacitance to noncritical net neighbors.

Syntax

```
CONLY_NETS: list_of_nets
```

Arguments

Argument	Description
list_of_nets	List of nets; wildcards can be used. (default: none)

Description

The CONLY_NETS command reports cross-capacitance to noncritical net neighbors. If the COUPLE_TO_GROUND: YES command is used, CONLY_NETS has no effect. If the COUPLE_TO_GROUND: NO command is used, the NETS command controls the behavior.

Examples

In the following example, CONLY_NETS has no effect and all nets are written to the netlist:

```
COUPLE_TO_GROUND: NO
NETS: *
```

In the following example, only net_a is extracted, including the coupling to net_b:

```
COUPLE_TO_GROUND: NO
NETS: net_a
CONLY_NETS: net_b
```

The previous example results in the following output:

```
*|NET net_a
…
*CAP
1 net_a:23 net_b 1.32
2 net_a:3433 net_b 12.46
```

See Also

- COUPLE_TO_GROUND
- NETLIST_COUPLE_UNSELECTED_NETS
- NETS

Feedback

CONSIDER_CENTER_ENCLOSURE_FOR_VIA_MERGE

Specifies how to merge vias based on the location of the merged via. Valid only for transistor-level flows.

Syntax

```
CONSIDER_CENTER_ENCLOSURE_FOR_VIA_MERGE: YES | NO
```

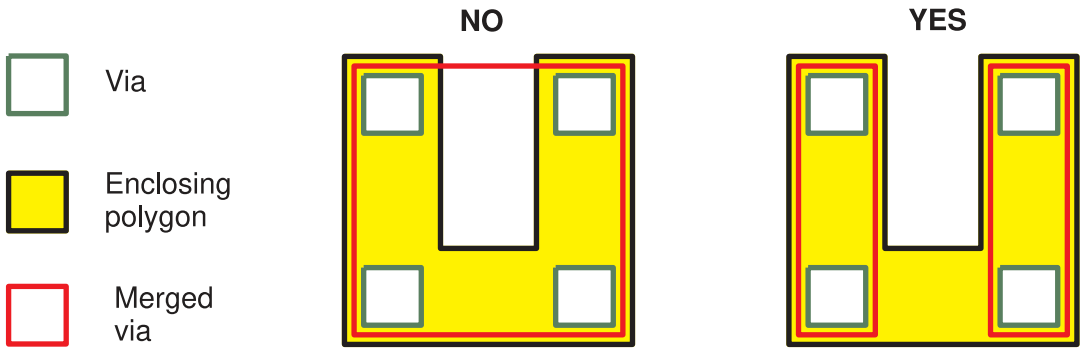
Arguments

Argument	Description
YES	Merge vias only if the center of the merged via is enclosed by the enclosing polygon.
NO (default)	Merge vias if they are enclosed by the same enclosing polygon.

Description

The CONSIDER_CENTER_ENCLOSURE_FOR_VIA_MERGE command controls via merging based on the location of the merged via with respect to enclosing conductors. By default, the tool does not evaluate whether the center of the merged via occurs inside a conductor enclosing polygon. For example, in Figure 180, the default setting of NO results in merging all four vias, but the center of the merged via falls outside the conductor polygons.

Figure 180 Effect of the Command Setting on Via Merging



Setting the CONSIDER_CENTER_ENCLOSURE_FOR_VIA_MERGE command to YES results in merging the vias in sets of two. In this case, the centers of the merged vias fall inside the conductor polygons.

Feedback

See Also

- MERGE_VIAS_IN_ARRAY

Feedback

CONTEXT_DEF_FILE

Specifies a DEF file that contains the redistribution layers. Valid only for LEF/DEF flows.

Syntax

```
CONTEXT_DEF_FILE: context_file
```

Arguments

Argument	Description
context_file	DEF file containing redistribution layers Default: none

Description

A redistribution layer (RDL) is a top-level metal layer that makes the input/output pads of a circuit available in other locations to facilitate packaging options such as flip-chip or wafer-on-wafer packaging. For hierarchical extraction, taking redistribution layers into account improves accuracy.

For gate-level LEF/DEF designs, use the `CONTEXT_DEF_FILE` command to specify the DEF file that contains the RDL information.

The content of this file must meet the following requirements:

- All of the shapes in this file should use a coordinate system in relation to the origin of the top block.
- The shapes must be grounded.
- The block name must be the same as the top-level design name.
- The data should be a flattened design. Therefore the COMPONENTS section is ignored.

See Also

- CONTEXT_GDS_FILE
- CONTEXT_OASIS_FILE

Feedback

CONTEXT_GDS_FILE

Specifies a GDSII file that contains the redistribution layers. Valid only for gate-level flows.

Syntax

```
CONTEXT_GDS_FILE: context_file
```

Arguments

Argument	Description
context_file	GDSII file containing redistribution layers Default: none

Description

A redistribution layer (RDL) is a top-level metal layer that makes the input/output pads of a circuit available in other locations to facilitate packaging options such as flip-chip or wafer-on-wafer packaging. For hierarchical extraction, taking redistribution layers into account improves accuracy.

For gate-level GDSII flows, use the `CONTEXT_GDS_FILE` command to specify the GDSII file that contains the RDL information.

The content of this file must meet the following requirements:

- All of the shapes in this file should use a coordinate system in relation to the origin of the top block.
- The shapes must be grounded.
- The block name must be the same as the top-level design name.
- The data should be a flattened design.

When you use this command, you must also use the `GDS_LAYER_MAP_FILE` command to provide the layer mapping information. The layers must be set to `GROUNDED`.

See Also

- CONTEXT_DEF_FILE
- CONTEXT_OASIS_FILE
- GDS_LAYER_MAP_FILE
- The StarXtract -gdscheck Option

Feedback

CONTEXT_OASIS_FILE

Specifies an OASIS file that contains the redistribution layers. Valid only for OASIS flows.

Syntax

```
CONTEXT_OASIS_FILE: context_file
```

Arguments

Argument	Description
context_file	OASIS file containing redistribution layers Default: none

Description

A redistribution layer (RDL) is a top-level metal layer that makes the input/output pads of a circuit available in other locations to facilitate packaging options such as flip-chip or wafer-on-wafer packaging. For hierarchical extraction, taking redistribution layers into account improves accuracy.

For gate-level OASIS flows, use the `CONTEXT_OASIS_FILE` command to specify the OASIS file that contains the RDL information.

The content of this file must meet the following requirements:

- All of the shapes in this file should use a coordinate system in relation to the origin of the top block.
- The shapes must be grounded.
- The block name must be the same as the top-level design name.
- The data should be a flattened design. Therefore the COMPONENTS section is ignored.

When you use this command, you must also use the `OASIS_LAYER_MAP_FILE` command to provide the layer mapping information. The layers must be set to `GROUNDED`.

See Also

- CONTEXT_DEF_FILE
- CONTEXT_GDS_FILE
- OASIS_LAYER_MAP_FILE

Feedback

CONVERT_DIODE_TO_PARASITIC_CAP

Specifies the extraction of parasitic properties of antenna diode structures. Valid only for transistor-level flows.

Syntax

```
CONVERT_DIODE_TO_PARASITIC_CAP: model_name area_coeff perimeter_coeff
```

Arguments

Argument	Description
model_name	Antenna diode model name Default: none
area_coeff	Area capacitance coefficient Units: F/m^2 Default: none
perimeter_coeff	Perimeter capacitance coefficient Units: F/m Default: none

Description

Use the `CONVERT_DIODE_TO_PARASITIC_CAP` command to extract parasitic properties of antenna diode structures.

Errors

Error messages are issued when

- The model name is not found
- The capacitance coefficients are not realistic values such as negative numbers
- Device properties are not found in the input data

Examples

```
CONVERT_DIODE_TO_PARASITIC_CAP: NpParaDiode 1e-15 1e-16
```

Feedback

CONVERT_WARNING_TO_ERROR

Specifies warning messages for which to halt execution instead of continuing.

Syntax

```
CONVERT_WARNING_TO_ERROR: ID_1 ID_2 ...
```

Arguments

Argument	Description
ID_1, ID_2, ...	Affected message ID numbers, separated by spaces (default: none)

Description

The `CONVERT_WARNING_TO_ERROR` command allows you to choose to halt extraction when the StarRC tool encounters specified warning conditions.

These features apply only to SX, EX, and GRD messages, which are messages issued by the StarRC tool. During an extraction run, messages with different prefixes might appear.

Examples

The following command causes the run to stop if either a SX-2549 or EX-269 warning occurs:

```
CONVERT_WARNING_TO_ERROR: SX-2549 EX-269
```

Feedback

CORNERS_FILE

Specifies a file that contains the definitions of corners available for extraction in the simultaneous multicorner flow.

Syntax

```
CORNERS_FILE: corner_file_name
```

Arguments

Argument	Description
corner_file_name	Name of the file containing corner definitions

Description

The CORNERS_FILE command specifies a file that contains the corner definitions of all corners available to be extracted in the simultaneous multicorner (SMC) flow. To specify which of the defined corners to extract, use the SELECTED_CORNERS command in the StarRC command file. These commands have an effect only if the SMC flow is enabled.

The CORNERS_FILE and STAR_DIRECTORY command arguments must follow these naming conventions:

- If the STAR_DIRECTORY command specifies a relative path, you can use either a relative path or an absolute path in the CORNERS_FILE command. For example:

  ```
  STAR_DIRECTORY: star_work
  CORNERS_FILE: smc_config
  ```

- If the STAR_DIRECTORY command specifies an absolute path, you must use an absolute path in the CORNERS_FILE command. For example:

  ```
  STAR_DIRECTORY: /tmp/star
  CORNERS_FILE: /remote/.../work_directory/smc_config
  ```

You can use the following commands in the corners file for all design databases:

```
CORNER_NAME: name_of_corner
TCAD_GRD_FILE: nxtgrd_path_and_file_name
OPERATING_TEMPERATURE: temperature_in_Celsius

(optional) MAPPING_FILE: map_file_name
(optional) VIA_COVERAGE_OPTION_FILE: via_file
(optional) DENSITY_OUTSIDE_BLOCK: density_value
(optional) 3D_IC_SUBCKT_FILE: filename
```

If you specify the 3D_IC_SUBCKT_FILE command in the StarRC command file and

- Specify the 3D_IC_SUBCKT_FILE command in the corner file, the tool considers corners specified in the corner file only
- Do not specify the 3D_IC_SUBCKT_FILE command in the corner file, the tool considers corners specified in the StarRC command file only

RC Scaling for Fusion Compiler or IC Compiler II Designs

The following commands can be used in the corners file only for NDM format designs created by the Fusion Compiler or IC Compiler II tool:

```
(optional) RES_SCALE: res_value
(optional) CAP_SCALE: cap_value
(optional) CC_SCALE: cc_value
```

The default for the scaling parameters is 1.0, which is equivalent to no scaling. The RES_SCALE factor applies to all resistors, excluding special resistors such as shorting resistors. The CC_SCALE factor applies to all coupling capacitances, excluding coupling capacitances to ground. The CAP_SCALE factor applies to all capacitances including total capacitance, excluding ground capacitances.

Coupling capacitances other than coupling capacitances to ground are subject to both the CC_SCALE and CAP_SCALE factors. Coupling capacitance to ground is adjusted to preserve the total capacitance of the node scaled by the CAP_SCALE value. If the capacitance to ground becomes negative after scaling, it is set to zero.

The scaling parameters are used in the following ways:

- During In-Design extraction in the Fusion Compiler or IC Compiler II tool

  The Fusion Compiler or IC Compiler II tool inserts the scaling commands into the corners file based on settings in the Fusion Compiler or IC Compiler II flow.

- During standalone StarRC extraction

  You can use the scaling commands in a corners file to compare standalone StarRC extraction to Fusion Compiler or IC Compiler II In-Design extraction.

Note:

You can use RC scaling in a standalone StarRC extraction run for comparison with IC Compiler II In-Design extraction runs. However, you should not use RC scaling for final signoff extraction runs.

Examples

Standard netlist output

The corners file CF.txt contains the following commands:

```
CORNER_NAME: NOM_T1
TCAD_GRD_FILE: nominal.nxtgrd
```

```
OPERATING_TEMPERATURE: -25

CORNER_NAME: NOM_T2
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: 125

CORNER_NAME: RCMAX_T3
TCAD_GRD_FILE: rcmax.nxtgrd
OPERATING_TEMPERATURE: 25
```

The command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: CF.txt
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
```

This extraction generates three output netlists whose names are star_NOM_T2.spf, star_NOM_T2.spf, and star_RCMAX_T3.spf. Each netlist is a standard netlist (subject to other netlist options such as including tail comments or selecting the format).

See Also

- SELECTED_CORNERS
- SIMULTANEOUS_MULTI_CORNER
- OPERATING_TEMPERATURE
- MAPPING_FILE
- VIA_COVERAGE_OPTION_FILE
- DENSITY_OUTSIDE_BLOCK
- 3D_IC_SUBCKT_FILE
- Simultaneous Multicorner Extraction

Feedback

COUPLE_NONCRITICAL_NETS

Reports the actual net names when coupling to material outside of the primary extraction cell.

Syntax

```
COUPLE_NONCRITICAL_NETS: cell_list
```

Arguments

Argument	Description
cell_list	A cell or a list of cells for reporting coupling capacitance outside the primary extraction cell Default: `!*`

Description

Reports the actual net names when coupling to material outside of the primary extraction cell. This command affects the child cells of `BLOCK` and the parent, sibling, and child cells of macros.

If you add a child cell to this list (that is, down from the primary cell), primary cell nets can couple to the real hierarchical net names in the child. The child cells are not included in the netlist and are floating nodes in the output SPEF file. If you add a parent or a sibling cell to this list, a special naming scheme is used to identify the coupling nodes. Instead of using full hierarchical names, the noncritical coupling nodes are named *cell_name/local_net_name*.

This command overrides the `ZONE_COUPLE_TO_NET` and `SKIP_CELLS_COUPLE_TO_NET` commands for the selected cells.

The *, ? and ! wildcards are acceptable. This command can be specified multiple times.

Note:

You should explicitly specify the cells in this list instead of using the asterisk (`*`) wildcard. Using a wildcard slows down runtime unnecessarily.

Examples

This example extracts an instance XU0 of cell MacroA in the context of the BLOCK TopBlock. MacroB is a sibling of MacroA, and SubMacroC is a child of MacroA. The

following example shows how to configure the related options to achieve the following result:

- Parent TopBlock and siblings MacroA and MacroB are coupled with special block and net names.
- Other siblings of MacroA, such as standard cells, are coupled with the net name ZONE.
- Child SubMacroC is coupled with real hierarchical net names, from the perspective of MacroA.
- Other subcells of MacroA are coupled with the net name LUMP.

```
BLOCK: TopBlock
COUPLE_NONCRITICAL_NETS: TopBlock MacroA MacroB SubMacroC
SKIP_CELLS_COUPLE_TO_NET: LUMP
ZONE_COUPLE_TO_NET: ZONE
```

The resulting SPEF netlist might report capacitors as follows:

```
*CAP
…
11 net1 TopBlock/clk 1.2e-13
12 net2 MacroA/net1 1e-16 // can couple to
   identical  // sibling
13 net3 MacroB/net2 1.3e-18
14 net1 instA/net1 8.2e-17 // couple to SubMacroC
   cell
…
*END
```

See Also

- BLOCK
- COUPLE_NONCRITICAL_NETS_PREFIX
- COUPLE_TO_GROUND
- NETLIST_FORMAT
- SKIP_CELLS_COUPLE_TO_NET
- ZONE_COUPLE_TO_NET

Feedback

COUPLE_NONCRITICAL_NETS_PREFIX

Specifies a prefix for the nets output by the COUPLE_NONCRITICAL_NETS command.

Syntax

```
COUPLE_NONCRITICAL_NETS_PREFIX: prefix
```

Arguments

Argument	Description
prefix	Prefix string Default: SYNOPSYS_INCONTEXT_

Description

Changes the prefix used by the COUPLE_NONCRITICAL_NETS command flow for nets, which must be made unique to preserve independent names.

From the specified BLOCK down the hierarchy, this command applies the prefix to interconnect or port nets of selected COUPLE_NONCRITICAL_NETS and SKIP_CELLS. From a specified macro up the hierarchy, the prefix is applied to all names in the external environment. For example, *instance/prefix_netname* is applied for all noncritical nets.

If you do not specify any value, the default is SYNOPSYS_INCONTEXT_. If you specify NONE (not case-sensitive), an empty prefix is used such that the coupling netname is *instance/netname*.

This command is ignored if you do not specify the COUPLE_NONCRITICAL_NETS command.

See Also

- COUPLE_NONCRITICAL_NETS

Feedback

COUPLE_NONCRITICAL_NETS_SUBNODE_SUFFIX

Specifies a netlist delimiter between the netname and suffix.

Syntax

```
COUPLE_NONCRITICAL_NETS_SUBNODE_SUFFIX: netlistDelimiter
```

Arguments

Argument	Description
netlistDelimiter	Netlist delimiter to be inserted between the netname and suffix Default: an empty string

Description

The COUPLE_NONCRITICAL_NETS_SUBNODE_SUFFIX command specifies a netlist delimiter between the netname and suffix. For example,

instance/prefix_netname_netlistDelimiter_suffix

This command only works for cell-level extraction.

Retaining coupling capacitances between the top-level parent routing and SKIP_CELLS child net routing exists for the Milkyway flow using the SPEF netlist format.

Examples

```
MY_SUB_GROUP_1/SYNOPSYS_INCONTEXT_n192:1
```

See Also

- COUPLE_NONCRITICAL_NETS
- NETLIST_FORMAT
- RING_AROUND_THE_BLOCK
- SKIP_CELLS_COUPLE_TO_NET
- ZONE_COUPLE_TO_NET

Feedback

COUPLE_TO_GROUND

Specifies whether coupling capacitances are retained or lumped to ground.

Syntax

```
COUPLE_TO_GROUND: YES | NO | YES RETAIN_GATE_CONTACT_COUPLING
                | YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING
```

Arguments

Argument	Description
YES (default)	Grounds all coupling capacitors and applies scaling factor
NO	Retains coupling capacitors if they meet threshold settings
YES RETAIN_GATE_CONTACT_COUPLING	Retains coupling capacitances between the gate and the via or trench contact. Gate, trench contact, and diffusion layers are identified by the LAYER_TYPE declaration in the ITF file. You must also set the EXTRACT_VIA_CAPS command to YES. Coupling capacitance thresholds do not apply. Valid for IC Validator, Hercules, and Calibre flows.
YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING	Retains coupling capacitances between the gate and the diffusion layer in addition to the coupling capacitances between the gate and the via or trench contact. Gate, trench contact, and diffusion layers are identified by the LAYER_TYPE declaration in the ITF file. You must also set the EXTRACT_VIA_CAPS command to YES and the IGNORE_CAPACITANCE command to NONE, DIFF, or ALL RETAIN_GATE_DIFFUSION_COUPLING. Coupling capacitance thresholds do not apply. Valid for IC Validator, Hercules, and Calibre flows.

Description

By default, or if you set the COUPLE_TO_GROUND command to YES, the StarRC tool grounds all coupling capacitors.

If you set the COUPLE_TO_GROUND command to NO, the tool evaluates the coupling capacitances based on the settings of the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands. As a result, coupling capacitances might be retained or grounded, depending on the capacitance value.

The tool scales all grounded capacitances by the factor specified by the COUPLING_MULTIPLIER command, which approximates the crosstalk effects that are lost by grounding the coupling capacitors.

Feedback

You can retain gate-to-contact capacitance by using either of the following methods:

- Set the COUPLE_TO_GROUND command to YES RETAIN_GATE_CONTACT_COUPLING (or to YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING) and the EXTRACT_VIA_CAPS command to YES. In this case, the gate-to-contact capacitance is retained, regardless of the settings of the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands.
- Set the RETAIN_GATE_CONTACT_COUPLING command to YES and the EXTRACT_VIA_CAPS command to YES.
  - If the COUPLE_TO_GROUND command is set to NO, the gate-to-contact capacitance is retained only if the capacitance meets the thresholds specified by the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands.
  - If the COUPLE_TO_GROUND command is set to YES, the gate-to-contact capacitance is always retained.

The RETAIN_GATE_CONTACT_COUPLING command takes precedence. If you set the RETAIN_GATE_CONTACT_COUPLING command to NO and the COUPLE_TO_GROUND command to YES RETAIN_GATE_CONTACT_COUPLING, the COUPLE_TO_GROUND setting is reset to YES and the gate-to-contact capacitance is not retained.

By default, the StarRC tool does not retain coupling capacitances for power nets. To retain the gate-to-contact capacitance for power nets, use one of these command settings:

- Set the POWER_EXTRACT command to YES to retain the gate-to-contact capacitances for all signal and power nets.
- Set the POWER_EXTRACT command to DEVICE_LAYERS to retain the gate-to-contact capacitances for all signal nets and all power nets whose layers are specified with the device_layers keyword in a conducting_layers statement in the mapping file.

Note:

If the device SPICE models already contain gate-to-contact capacitances, you would typically avoid capacitance double counting by setting the EXTRACT_VIA_CAPS command to YES IGNORE_GATE_CONTACT_COUPLING. In this case, the tool does not extract the gate-to-contact capacitances; therefore, the commands discussed in this section have no effect on these capacitances.

See Also

- COUPLING_MULTIPLIER
- COUPLING_ABS_THRESHOLD
- COUPLING_REL_THRESHOLD
- EXTRACT_VIA_CAPS

- IGNORE_CAPACITANCE
- RETAIN_GATE_CONTACT_COUPLING

Feedback

COUPLE_TO_PCELL_PINS

Specifies the coupling to pins of a parameterized cell (PCell). Valid only for transistor-level flows.

Syntax

```
COUPLE_TO_PCELL_PINS: NO | YES | YES KEEP_CG | YES IGNORE_CG
            | YES AUTOMATIC_CG_HANDLING
```

Arguments

Argument	Description
NO (default)	Ignores the coupling of PCell pins to all material outside the PCell
YES	Extracts the coupling of PCell pins to adjacent PCell pins and to overhead metal routing. Keeps coupling between the PCell pins and ground planes on SUBSTRATE layers.
YES KEEP_CG (same as YES)	Extracts the coupling of PCell pins to adjacent PCell pins and to overhead metal routing. Keeps coupling between the PCell pins and ground planes on SUBSTRATE layers.
YES IGNORE_CG	Extracts the coupling of PCell pins to adjacent PCell pins and to overhead metal routing. Ignores couplings between PCell pins and ground planes on SUBSTRATE layers.
YES AUTOMATIC_CG_HANDLING	Extracts coupling of PCell pins to adjacent PCell pins and to overhead routing. Extracts coupling between PCell pins and ground planes on SUBSTRATE layers only if SUBSTRATE polygons are not inside the PCell.

Description

The COUPLE_TO_PCELL_PINS command is a global command that controls whether the StarRC tool extracts PCell pin coupling capacitance to adjacent PCell pins and ground. The command setting applies to all PCells except for specific PCells named in an instance of the COUPLE_TO_SPECIFIED_PCELL_PINS command with a different setting.

When you set the COUPLE_TO_PCELL_PINS command to YES AUTOMATIC_CG_HANDLING, the tool retains coupling capacitances to outside ground planes only if there are no substrate polygons in the cell. If substrate polygons are present in the cell, the tool drops the capacitances to outside ground planes.

However, an exception can occur if the PCell includes floating metal fill. In this case, the tool reports coupling capacitance between the PCell pins and the substrate even if the YES AUTOMATIC_CG_HANDLING setting is specified. The characterized PCell models do not typically include coupling between metal fill and the substrate or coupling between

metal fill and the PCell pins. Therefore the StarRC tool captures the coupling between the substrate and the PCell pins that happens through the metal fill.

Examples

In the following example, the StarRC tool extracts coupling capacitance between adjacent PCell pins and couplings between PCell pins and ground, for all PCells in the design:

```
COUPLE_TO_PCELL_PINS: YES KEEP_CG
```

See Also

- COUPLE_TO_SPECIFIED_PCELL_PINS
- SKIP_PCELLS
- PCELL_EXTRACTION_FILE

Feedback

COUPLE_TO_SPECIFIED_PCELL_PINS

Specifies the coupling of parameterized cell (PCell) pins for specific PCells. Valid only for transistor-level flows.

Syntax

```
COUPLE_TO_SPECIFIED_PCELL_PINS: NO | YES | YES KEEP_CG | YES IGNORE_CG
            | YES AUTOMATIC_CG_HANDLING | YES KEEP_ALL_CG cell_name
```

Arguments

Argument	Description
NO (default)	Ignores the coupling of PCell pins to all material outside the PCell.
YES	Extracts the coupling of PCell pins to adjacent PCell pins and to overhead metal routing. Keeps coupling between the PCell pins and ground planes on SUBSTRATE layers.
YES KEEP_CG (same as YES)	Extracts the coupling of PCell pins to adjacent PCell pins and to overhead metal routing. Keeps coupling between the PCell pins and ground planes on SUBSTRATE layers and outside of the PCell.
YES IGNORE_CG	Extracts the coupling of PCell pins to adjacent PCell pins and to overhead metal routing. Ignores couplings between PCell pins and ground planes on SUBSTRATE layers.
YES AUTOMATIC_CG_HANDLING	Extracts coupling of PCell pins to adjacent PCell pins and to overhead routing. Extracts coupling between PCell pins and ground planes on SUBSTRATE layers only if SUBSTRATE polygons are not inside the PCell.
YES KEEP_ALL_CG	Extracts the coupling of PCell pins to adjacent PCell pins and to overhead metal routing. Keeps coupling between the PCell pins and ground planes on SUBSTRATE layers inside and outside the PCell.
cell_name	The cell to which the command applies. Wildcards * and ? are accepted.

Description

The COUPLE_TO_SPECIFIED_PCELL_PINS command is a cell-specific command that controls whether the StarRC tool extracts PCell pin coupling capacitance to adjacent PCell pins and ground.

This command requires a setting and a cell name as arguments. The settings are the same as for the COUPLE_TO_PCELL_PINS command, which is a global command that applies to all PCells except for cells named in an instance of the COUPLE_TO_SPECIFIED_PCELL_PINS command.

Feedback

The following usage notes apply:

- You can use the * and ? wildcards in the cell name. The cell name is case-sensitive if the CASE_SENSITIVE command is set to YES (the default).
- You can use this command multiple times in a StarRC command file. If specific PCells are named multiple times, later instances of this command overwrite earlier instances.
- All cells named in this command must also be named in a SKIP_PCELLS command. If a cell name does not appear in both commands, the tool issues a warning message and ignores the COUPLE_TO_SPECIFIED_PCELL_PINS command setting for that cell.

Figure 181 Handling Ground Capacitance With and Without SUBSTRATE Layers Inside PCELL

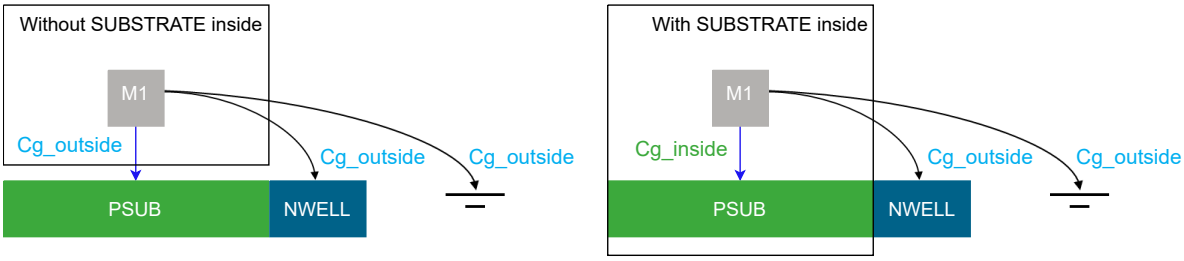


Figure 181 demonstrates how the StarRC tool handles ground capacitance using the following options, with and without SUBSTRATE layer inside the PCELL:

- The KEEP_CG option (the default) keeps Cg_outside and drops Cg_inside.
- The IGNORE_CG option drops both Cg_outside and Cg_inside.
- The AUTOMATIC_CG_HANDLING option drops Cg_outside if SUBSTRATE layer is inside the PCELL and drops Cg_inside.
- The KEEP_ALL_CG option keeps both Cg_outside and Cg_inside.

Examples

In the following example, cells A, B, and C are PCells. The global setting for coupling is YES. The first COUPLE_TO_SPECIFIED_PCELL_PINS command sets the coupling for cell A to NO. The second COUPLE_TO_SPECIFIED_PCELL_PINS command sets the coupling for cell B to YES IGNORE_CG. As a result, all three cells have different settings.

```
SKIP_PCELLS: A B C
COUPLE_TO_PCELL_PINS: YES
COUPLE_TO_SPECIFIED_PCELL_PINS: NO A
COUPLE_TO_SPECIFIED_PCELL_PINS: YES IGNORE_CG B
```

If the following command appears later in the StarRC command file, the coupling for cell A is overwritten and becomes `YES KEEP_CG`.

```
COUPLE_TO_SPECIFIED_PCELL_PINS: YES KEEP_CG A
```

If the following command appears later in the StarRC command file, the cell-specific commands take priority. The setting for cell A is not changed and continues to be `YES KEEP_CG`. The setting for cell B is not changed and continues to be `YES IGNORE_CG`. The setting for cell C, which has not been included in any cell-specific commands, changes from `YES` to `NO`.

```
COUPLE_TO_PCELL_PINS: NO
```

The following command specifies the coupling for the PCells (my_pcell*) to `YES KEEP_ALL_CG`.

```
COUPLE_TO_SPECIFIED_PCELL_PINS: YES KEEP_ALL_CG my_pcell*
```

See Also

- SKIP_PCELLS
- COUPLE_TO_PCELL_PINS
- PCELL_EXTRACTION_FILE

Feedback

COUPLING_ABS_THRESHOLD

Specifies an absolute threshold for grounding coupling capacitors.

Syntax

```
COUPLING_ABS_THRESHOLD: threshold
```

Arguments

Argument	Description
threshold	Absolute threshold for grounding coupling capacitors Units: farads (F) Default: 3e-15

Description

Specifies an absolute threshold for grounding coupling capacitors.

By default, coupling capacitors are grounded if both of the following conditions are met:

- The ratio of coupling capacitance to each individual net's total capacitance is less than the value specified by the `COUPLING_REL_THRESHOLD` command.
- The coupling capacitance is less than the value specified by the `COUPLING_ABS_THRESHOLD` command.

However, if you set the `COUPLING_THRESHOLD_OPERATION` command to `OR`, the coupling capacitance is grounded if either of the two conditions is satisfied.

See Also

- COUPLING_REL_THRESHOLD
- COUPLING_THRESHOLD_OPERATION

Feedback

COUPLING_MULTIPLIER

Specifies a scaling factor to be applied for transferring coupling capacitances to ground.

Syntax

```
COUPLING_MULTIPLIER: value
```

Arguments

Argument	Description
value	Floating-point number greater than zero Default: 1.0

Description

Applies a scaling factor when transferring coupling capacitances to ground. This command is used primarily to scale parasitic capacitances for crosstalk effects.

Examples

```
COUPLING_MULTIPLIER: 6
```

See Also

- COUPLE_TO_GROUND

Feedback

COUPLING_REL_THRESHOLD

Specifies the ratio of coupling capacitance to total capacitance used to determine whether to ground coupling capacitors.

Syntax

```
COUPLING_REL_THRESHOLD: threshold
```

Arguments

Argument	Description
threshold	Floating-point number between 0 and 1 Default: 0.03

Description

Specifies the ratio of coupling capacitance to total capacitance used as a threshold for grounding coupling capacitors.

Coupling capacitors are grounded if both of the following conditions are met:

- The ratio of coupling capacitance to each individual net's total capacitance is less than the value specified by the `COUPLING_REL_THRESHOLD` command.
- The coupling capacitance is less than the value specified by the `COUPLING_ABS_THRESHOLD` command.

However, if you set the `COUPLING_THRESHOLD_OPERATION` command to `OR`, the coupling capacitance is grounded if either of the two conditions is satisfied.

See Also

- COUPLING_ABS_THRESHOLD
- COUPLING_THRESHOLD_OPERATION

Feedback

COUPLING_REPORT_FILE

Generates a report listing the coupling capacitance by net.

Syntax

```
COUPLING_REPORT_FILE: file
```

Arguments

Argument	Description
file	File name for the report Default: none

Description

Generates a report listing the coupling capacitance by net after smart decoupling. The report is sorted by the percentage of coupling capacitance to total capacitance for the net. The report uses the following format:

Cc/Ct *100 Cc victim_net aggressor_net

The report contains the number of entries indicated by the COUPLING_REPORT_NUMBER command.

The total net capacitance used for the coupling percentage calculation is the net capacitance that is used for smart decoupling. It does not include loading pin capacitors and intranet coupling capacitors (same net coupling).

Examples

```
* 1000 worst couplings in descending order
* ratio(%) coupling victim aggressor
30.00 3e-15 net1 net2
20.00 2e-15 net3 net2
10.00 3e-15 net2 net1
```

See Also

- COUPLING_REPORT_NUMBER

Feedback

COUPLING_REPORT_NUMBER

Specifies the number of nets reported in the coupling report file.

Syntax

```
COUPLING_REPORT_NUMBER: no_of_nets
```

Arguments

Argument	Description
no_of_nets	Integer number of nets for which to report coupling capacitors Default: 1000

Description

Controls the size of the coupling capacitance report file by limiting the number of nets in the report.

Examples

```
COUPLING_REPORT_NUMBER: 5
```

See Also

- COUPLING_REPORT_FILE

Feedback

COUPLING_THRESHOLD_OPERATION

Specifies the use of AND filtering or OR filtering of coupling thresholds.

Syntax

```
COUPLING_THRESHOLD_OPERATION: AND | OR
```

Arguments

Argument	Description
`AND` (default)	AND filtering
`OR`	OR filtering

Description

The following conditions are checked to determine whether a coupling capacitance Cc(net1-net2) should be decoupled (grounded):

Condition1: Cc(net1-net2) < COUPLING_ABS_THRESHOLD

Condition2: Cc(net1-net2) < COUPLING_REL_THRESHOLD * TCAP_net1

Condition3: Cc(net1-net2) < COUPLING_REL_THRESHOLD * TCAP_net2

The `COUPLING_THRESHOLD_OPERATION` command specifies the use of AND filtering or OR filtering, as follows:

- When AND filtering is specified, a coupling capacitance is decoupled if the following operation is true:

  Condition1 AND (Condition2 AND Condition3)

- When OR filtering is specified, a coupling capacitance is decoupled if the following operation is true:

  Condition1 OR (Condition2 AND Condition3)

See Also

- COUPLING_ABS_THRESHOLD
- COUPLING_REL_THRESHOLD

Feedback

DEBUG_MILKYWAY_DATABASE

Specifies the name of a Milkyway database to use for debugging opens and shorts.

Syntax

```
DEBUG_MILKYWAY_DATABASE: mw_lib
```

Arguments

Argument	Description
mw_lib	A Milkyway database name Default: none

Description

The `DEBUG_MILKYWAY_DATABASE` command provides the name of a Milkyway database into which the StarRC tool writes data for debugging opens and shorts. If a database with this name already exists, it is overwritten. The `DEBUG_MILKYWAY_DATABASE` command is used only in a short StarRC command file referenced by the `StarXtract` command with one of the `-Display`, `-Display_mf`, or `-Display short_regions` options.

You can save the following types of data for debugging:

- Critical net (signal net) opens and shorts between critical nets, by using the `-Display` option
- Shorts between critical nets and metal fill, by using the `-Display_mf` option
- Shorts between critical nets and noncritical polygons, by using the `-Display short_regions` option

This feature is available for gate-level flows that do not use the field solver. Nets are displayed in mask layout dimensions after the application of any half-node scale factors.

The `StarXtract` command used with one of the display options does not perform extraction or create an output netlist. These options require a simple StarRC command file to generate the debugging database. The command file must contain one of the `DEBUG_NDM_DATABASE` or `DEBUG_MILKYWAY_DATABASE` commands to specify the name of the database. The debugging database format is independent of the format of the original design.

To investigate opens and shorts, follow this procedure:

1. Perform a standard StarRC extraction run.
2. Create a simple StarRC command file for the purpose of visualizing opens and shorts. The following example is a command file named star_cmd_debug. Use the `NETS` command to select specific nets to view; selecting all nets is not recommended.

   ```
   DEBUG_MILKYWAY_DATABASE: my_library
   STAR_DIRECTORY: star
   NETS: net1 net2 net3
   ```

3. (Optional) If you plan to examine shorts between critical nets and noncritical polygons with the `-Display short_regions` option, include the following command in the command file:

   ```
   ENHANCED_SHORT_REPORTING: COMPLETE
   ```

4. If the design contains more than 256 layers, use the `remove_unused` option to reduce the number of layers in the design. Otherwise, some layers might not be visible in the layout viewer. The `remove_unused` option must appear at the end of the command, as follows:

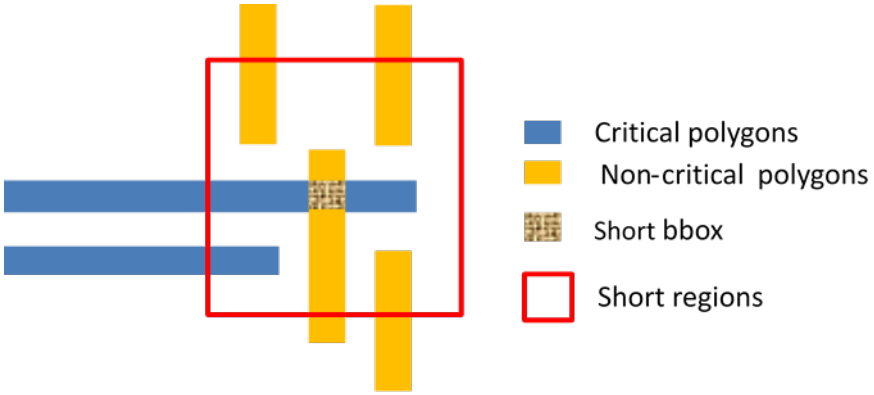
   ```
   % StarXtract -Display short_regions star_cmd_debug remove_unused
   ```

5. Invoke the StarRC tool with one of the display options and the name of the special command file. For example:

   ```
   % StarXtract -Display short_regions star_cmd_debug
   ```

   The StarRC tool saves a region of the design expanded around the site of each detected short. Figure 182 illustrates the saved region for the `-Display short_regions` option.

Figure 182 Region Saved Around a Short



6. Open a layout viewer to examine the new database.

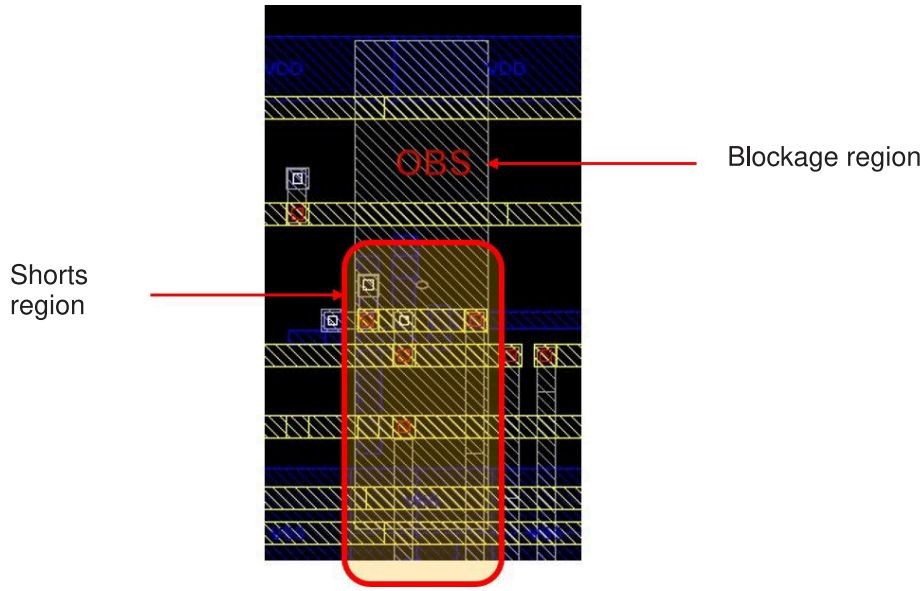
Feedback

Figure 183 is an example of a net identified as an open by the StarRC tool. Examination reveals that a via is missing between the M1 and M2 layers.

Figure 183 View of Signal Net Open

Figure 184 is an example of a region where signal nets are shorted to a blockage region.

Figure 184 Example View of Noncritical Short Polygons



Feedback

See Also

- DEBUG_NDM_DATABASE
- NETS
- Display Options for Debugging Opens and Shorts

Feedback

DEBUG_NDM_DATABASE

Specifies the name of an NDM format database to use for debugging opens and shorts.

Syntax

```
DEBUG_NDM_DATABASE: ndm_lib
```

Arguments

Argument	Description
ndm_lib	An NDM format database name Default: none

Description

The `DEBUG_NDM_DATABASE` command provides the name of an NDM format database into which the StarRC tool writes data for debugging opens and shorts. If a database with this name already exists, it is overwritten. The `DEBUG_NDM_DATABASE` command is used only in a short StarRC command file referenced by the `StarXtract` command with one of the `-Display`, `-Display_mf`, or `-Display short_regions` options.

You can save the following types of data for debugging:

- Critical net (signal net) opens and shorts between critical nets, by using the `-Display` option
- Shorts between critical nets and metal fill, by using the `-Display_mf` option
- Shorts between critical nets and noncritical polygons, by using the `-Display short_regions` option

This feature is available for gate-level flows that do not use the field solver. Nets are displayed in mask layout dimensions after the application of any half-node scale factors.

The `StarXtract` command used with one of the display options does not perform extraction or create an output netlist. These options require a simple StarRC command file to generate the debugging database. The command file must contain one of the `DEBUG_NDM_DATABASE` or `DEBUG_MILKYWAY_DATABASE` commands to specify the name of the database. The debugging database format is independent of the format of the original design.

Feedback

To investigate opens and shorts, follow this procedure:

1. Perform a standard StarRC extraction run.
2. Create a simple StarRC command file for the purpose of visualizing opens and shorts. The following example is a command file named star_cmd_debug. Use the `NETS` command to select specific nets to view; selecting all nets is not recommended.

```
DEBUG_NDM_DATABASE: my_library
STAR_DIRECTORY: star
NETS: net1 net2 net3
```

3. (Optional) If you plan to examine shorts between critical nets and noncritical polygons with the `-Display short_regions` option, include the following command in the command file:

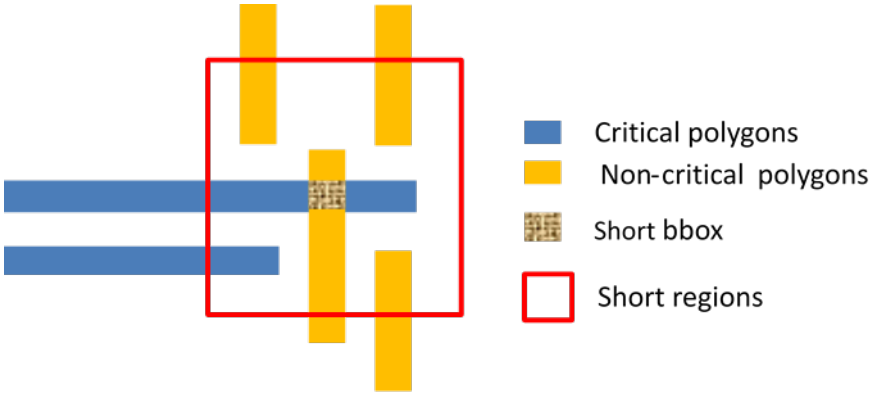
```
ENHANCED_SHORT_REPORTING: COMPLETE
```

4. Invoke the StarRC tool with one of the display options and the name of the special command file. For example:

```
% StarXtract -Display short_regions star_cmd_debug
```

   The StarRC tool saves a region of the design expanded around the site of each detected short. Figure 185 illustrates the saved region for the `-Display short_regions` option.

Figure 185 Region Saved Around a Short



5. Open a layout viewer to examine the new database.

Figure 186 is an example of a net identified as an open by the StarRC tool. Examination reveals that a via is missing between the M1 and M2 layers.

Feedback

Figure 186 View of Signal Net Open

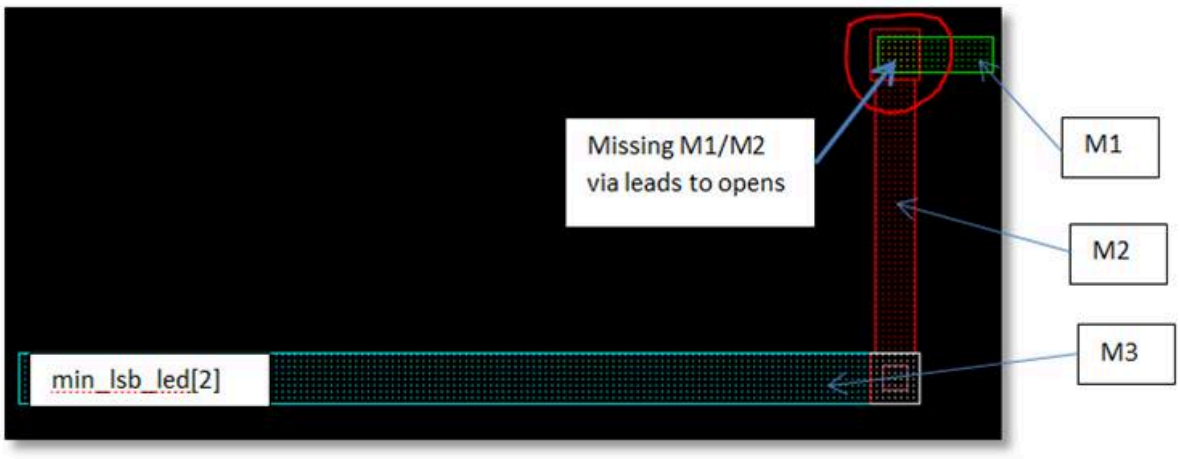
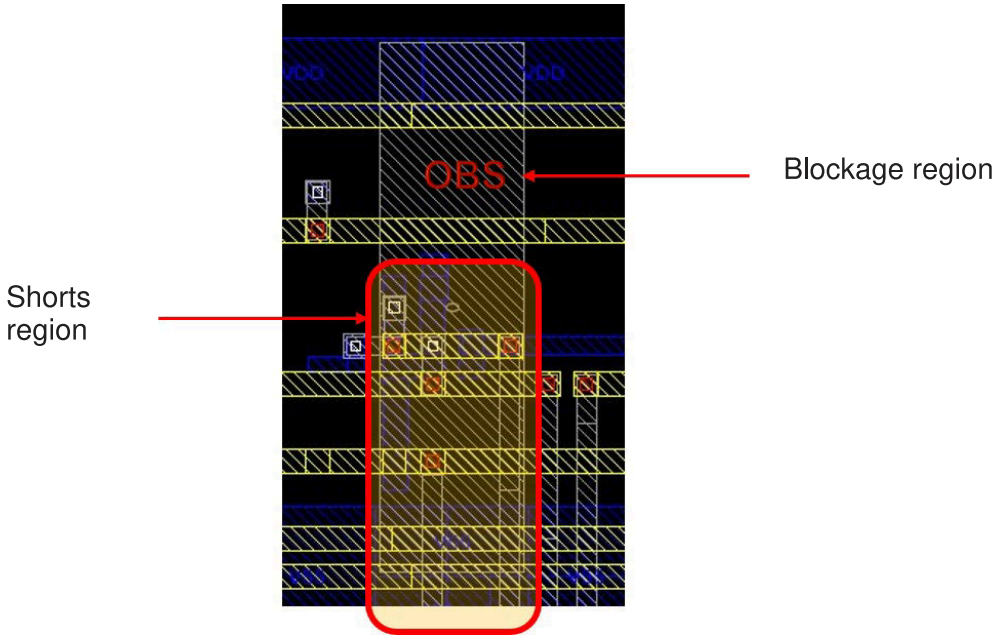


Figure 187 is an example of a region where signal nets are shorted to a blockage region.

Figure 187 Example View of Noncritical Short Polygons



Feedback

See Also

- DEBUG_MILKYWAY_DATABASE
- NETS
- Display Options for Debugging Opens and Shorts

Feedback

DEF_ATTRIBUTE_FROM_LEF

Assigns one or more attributes from a specific LEF file to a specific DEF macro.

Syntax

```
DEF_ATTRIBUTE_FROM_LEF: prop_list def_macro_name lef_file_name
```

Arguments

Argument	Description
prop_list	The attributes (properties) to be retrieved from the LEF file, separated by the vertical bar (|) character Valid values: WIDTH, WRONGDIRECTION, VIA, NDR
def_macro_name	The DEF macro name
lef_file_name	The LEF file from which to obtain the attribute information

Description

By default, the StarRC tool uses LEF attributes from the default LEF file for DEF macros. The DEF_ATTRIBUTE_FROM_LEF command obtains one or more LEF attributes from the specified LEF file and overwrites the defaults of those attributes for the specified DEF macro.

You can specify the DEF_ATTRIBUTE_FROM_LEF command multiple times in a command file. However, a specific DEF macro can obtain properties from only one LEF file.

The following four attributes can be obtained from the LEF file: WIDTH, VIA, NDR, and WRONGDIRECTION.

If you want to overwrite more than one attribute for a specific macro, you must specify all of the attributes in a single DEF_ATTRIBUTE_FROM_LEF command. Separate the attribute names with the vertical bar character (|) and no spaces. For example:

```
DEF_ATTRIBUTE_FROM_LEF: WIDTH|VIA|NDR macroA.def tech1.lef
```

When you use the the DEF_ATTRIBUTE_FROM_LEF command, a directory named DEFoverride is automatically created under the STAR directory. For every macro specified for an override, the tool creates a summary file listing all of the attributes that are overwritten from the specified LEF file.

To generate a report that includes all of the override information, set the DEF_OVERRIDE_REPORT_FILE command to a file name. The report file is located in the run directory and includes the macro DEF file name, the override LEF file name, and a list of the property types and property names that are replaced.

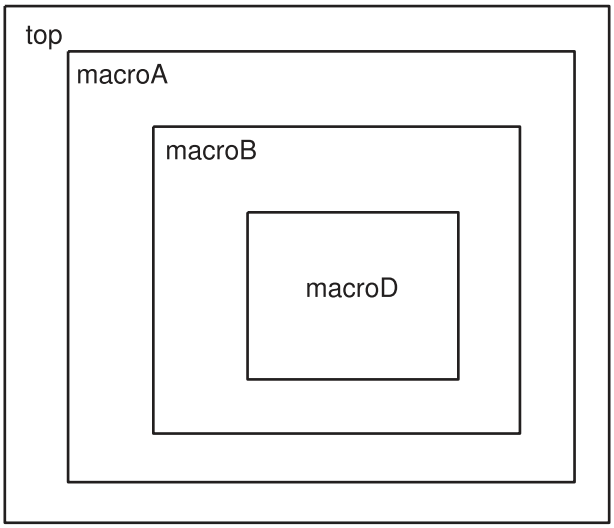
Feedback

Examples

The following commands assign the routing width from different LEF files to specific DEF macros. Figure 188 illustrates the relationship between the macros.

```
LEF_FILE: tech.lef
TOP_DEF_FILE: top.def
MACRO_DEF_FILE: macroA.def
MACRO_DEF_FILE: macroB.def
MACRO_DEF_FILE: macroD.def

DEF_ATTRIBUTE_FROM_LEF: WIDTH macroA tech1.lef
DEF_ATTRIBUTE_FROM_LEF: WIDTH macroD tech5.lef
```

Figure 188 Macros in the top.def File of a Hierarchical Design



The macros named top and macroB use the routing width from the default LEF file (tech.lef). MacroA and macroD use routing width values from LEF files tech1 and tech5, respectively, due to the use of the DEF_ATTRIBUTE_FROM_LEF command.

See Also

- DEF_OVERRIDE_REPORT_FILE
- DEF File Override Reports

Feedback

DEF_MASKSHIFT_CONSISTENCY_CHECK

Specifies whether to check LEF and DEF files for consistency of mask shift information.

Syntax

```
DEF_MASKSHIFT_CONSISTENCY_CHECK: YES | NO
```

Arguments

Argument	Description
YES (default)	Checks LEF and DEF files for mask shift consistency
NO	Disables mask shift consistency check

Description

By default, the StarRC tool checks the consistency of LEF and DEF files with respect to mask shift information for multiple patterning processes. If you use LEF and DEF files that originate from different sources, the consistency check helps to find errors.

The behavior is as follows:

- If the DEF file contains the MASKSHIFT keyword but the LEF files do not, the tool issues an error message and stops.
- If the DEF file does not contain the MASKSHIFT keyword but the LEF files do, the tool issues a warning message and continues.

To disable this check, set the DEF_MASKSHIFT_CONSISTENCY_CHECK command to NO. However, inconsistent files might lead to fatal errors or inaccurate results.

See Also

- LEF_FILE
- TOP_DEF_FILE
- MACRO_DEF_FILE
- The LEF/DEF Database Flow

Feedback

DEF_OVERRIDE_REPORT_FILE

Generates a report that lists all attributes that are overwritten with a value from a LEF file.

Syntax

```
DEF_OVERRIDE_REPORT_FILE: def_report
```

Arguments

Argument	Description
def_report	File name for the report Default: none

Description

The DEF_ATTRIBUTE_FROM_LEF command obtains one or more LEF attributes from a LEF file and overwrites the defaults of those attributes for a specified DEF macro.

To generate a report that includes the override information from all instances of the DEF_ATTRIBUTE_FROM_LEF command, set the DEF_OVERRIDE_REPORT_FILE command to a file name. The DEF override report file is located in the run directory and includes the macro DEF file name, the override LEF file name, and a list of the property types and property names that are replaced.

An example of the DEF override report is as follows:

```
MACRO Cell Name : MACROCELL1
LEF FILE : tech1.lef

VIA VIA12
VIA VIA23
VIA VIA56
NDR NDR_TEST
=========================================
MACRO Cell Name : periph_2
LEF FILE : .../tech/foundry_mxb2.lef

VIA VIA7045
NDR NDR_3p
```

See Also

- DEF_ATTRIBUTE_FROM_LEF
- DEF File Override Reports

Feedback

DEF_USE_PINS

Specifies whether to obtain pin information from DEF or LEF files.

Syntax

```
DEF_USE_PINS: YES | NO
```

Arguments

Argument	Description
YES (default)	Takes pin information first from the DEF file and then from the LEF file, if not present in the DEF file
NO	Takes pin information from the LEF file

Description

For a LEF/DEF flow, the DEF_USE_PINS command specifies whether to obtain pin information from the DEF file or the LEF file.

If the command is set to YES (the default), pin information is taken preferentially from the DEF file. Pins not found in the DEF file are taken from the LEF file. Conversely, if the DEF_USE_PINS command is set to NO, the pin information is taken only from the LEF file.

See Also

- LEF_FILE
- TOP_DEF_FILE
- MACRO_DEF_FILE
- The LEF/DEF Database Flow

Feedback

DEFAULT_CORNER

Defines the default corner to use as a reference corner for smart decoupling.

Syntax

```
DEFAULT_CORNER: corner_name | AUTOMATIC
```

Arguments

Argument	Description
corner_name	Sets the specified corner as the default corner. Note that the definition of the specified corner should be defined with the CORNERS_FILE command; however, the tool does not require this corner to be included with the SELECTED_CORNERS command.
AUTOMATIC	Selects automatically the worst coupling capacitance corner as the default corner from the list of corners defined with the CORNERS_FILE command. Make sure that you do not use AUTOMATIC as the corner name to define with CORNERS_FILE.
command not present	Sets the first corner as the default corner from the list of corners specified with the SELECTED_CORNERS command.

Description

This command defines the default corner to be used as a reference corner for smart coupling capacitance decoupling for all the corners specified with the SELECTED_CORNERS command in the simultaneous multicorner (SMC) flow.

Note:

The tool sets the first corner as the default corner from the list of corners specified with the SELECTED_CORNERS command if you do not use the DEFAULT_CORNER command.

When you use the DEFAULT_CORNER: AUTOMATIC command, the tool issues the following message:

```
Information: The tool sets the default corner: xxx in the simultaneous
 multicorner (SMC) flow (SX-2790)
```

Feedback

Examples

The following example shows how to specify the `DEFAULT_CORNER: AUTOMATIC` command for the tool to select automatically the worst coupling capacitance corner as the default corner from the list of corners specified in the CF.txt file:

```
DEFAULT_CORNER: AUTOMATIC

SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: CF.txt
```

See Also

- CORNERS_FILE
- SIMULTANEOUS_MULTI_CORNER
- Simultaneous Multicorner Extraction
- SELECTED_CORNERS
- COUPLING_THRESHOLD_OPERATION
- COUPLING_ABS_THRESHOLD
- COUPLING_REL_THRESHOLD

Feedback

DELETE_TRIVIAL_INSTANCE_PORTS

Deletes skip-cell (black-box cell) ports that are not connected to devices.

Syntax

```
DELETE_TRIVIAL_INSTANCE_PORTS: NO | YES
```

Arguments

Argument	Description
NO (default)	Does not delete trivial-instance ports
YES	Deletes trivial-instance ports

Description

For definition of trivial ports, see EXPLODE_TRIVIAL_INSTANCE_PORTS.

When you set the DELETE_TRIVIAL_INSTANCE_PORTS command to YES, the StarRC tool deletes the trivial-instance ports from the netlist. When a trivial-instance port is deleted, the polygons of the port are treated as non-critical. The resistance is extracted, but the capacitance is not extracted from the polygons. So, the neighbors of the noncritical polygons might observe the impact of capacitance.

See Also

- EXPLODE_TRIVIAL_INSTANCE_PORTS
- REMOVE_TRIVIAL_INSTANCE_PORTS

Feedback

DENSITY_BASED_THICKNESS

Enables the calculation of density and thickness variation during extraction.

Syntax

```
DENSITY_BASED_THICKNESS: YES | NO
```

Arguments

Argument	Description
YES (default)	Considers density-based thickness variation options as specified in the ITF file
NO	Does not consider density-based thickness options

Description

This command enables the calculation of density and thickness variation using the THICKNESS_VS_DENSITY or the POLYNOMIAL_BASED_THICKNESS_VARIATION commands in the ITF file.

If this command is not specified or is set to YES, the StarRC tool issues a warning if thickness variation commands are not specified in the ITF file. The tool does not issue a warning if you set the DENSITY_BASED_THICKNESS command to NO.

Examples

```
DENSITY_BASED_THICKNESS: YES
```

See Also

- NETS
- USE_SI_DENSITY
- POLYNOMIAL_BASED_THICKNESS_VARIATION
- THICKNESS_VS_DENSITY

Feedback

DENSITY_OUTSIDE_BLOCK

Specifies the pattern density outside the block, which affects the thickness variation and parasitic RC values.

Syntax

```
DENSITY_OUTSIDE_BLOCK: density_value
```

Arguments

Argument	Description
density_value	Pattern cell density; a floating-point number from 0.0 to 1.0, inclusive

Description

The DENSITY_OUTSIDE_BLOCK command defines the pattern density outside the block. The specified density is applied to all layers on which the StarRC tool performs density calculation.

For calculating the density of a polygon, the tool considers a 50-micron square window. If the polygon of interest is located near the edge of the block, the final density uses a weighted calculation that takes into account both the actual inside density and the outside density specified with the DENSITY_OUTSIDE_BLOCK command.

If the DENSITY_OUTSIDE_BLOCK command is not set, the tool extends the density inside the block to outside the block.

This command is effective only when you specify density-based thickness variation in the ITF file and include the DENSITY_BASED_THICKNESS: YES command in the StarRC command file.

This command has an effect only if the following conditions are met:

- The StarRC command file includes the DENSITY_BASED_THICKNESS: YES command.
- The ITF file contains one or more of the following density-based thickness variation specifications:
  - The THICKNES_VS_DENSITY command
  - The THICKNES_VS_DENSITY_AND_WIDTH command
  - The POLYNOMIAL_BASED_THICKNES_VARIATION command

You can optionally modify the ITF file to include either or both of the USE_SI_DENSITY command, which specifies whether to base the density on drawn or etched dimensions,

and the DENSITY_BOX_WEIGHTING_FACTOR command, which changes the dimensions of the analysis window.

Simultaneous Multicorner Extraction Support

For simultaneous multicorner extraction, you can define different pattern density values outside the design block for different corners by using the DENSITY_OUTSIDE_BLOCK command in the corners file. SMC extraction supports a maximum of 15 unique combinations of nxtgrd files and DENSITY_OUTSIDE_BLOCK specifications. This feature requires an Ultra+ license.

If the DENSITY_OUTSIDE_BLOCK command is not set for a corner, the tool applies the value specified by the DENSITY_OUTSIDE_BLOCK command in the StarRC command file. If the StarRC command file does not include a DENSITY_OUTSIDE_BLOCK command, the tool extends the density inside the block to outside the block.

Examples

The following command specifies a pattern density outside the block of 40 percent:

```
DENSITY_OUTSIDE_BLOCK: 0.40
```

See Also

- DENSITY_BASED_THICKNESS
- USE_SI_DENSITY
- THICKNESS_VS_DENSITY
- POLYNOMIAL_BASED_THICKNESS_VARIATION
- Simultaneous Multicorner Extraction

Feedback

DETECT_FUSE

Enables the detection and reporting of abutting metal polygons and other types of jog patterns.

Syntax

```
DETECT_FUSE: YES | NO
```

Arguments

Argument	Description
YES	Enables metal polygon overlap analysis
NO (default)	Disables metal polygon overlap analysis

Description

The DETECT_FUSE command detects instances of abutting metal polygons or conductor jog patterns that might pose electromigration risks. A fuse is a metal pattern in which an open circuit can be intentionally created by forcing current through a region of reduced width. Fuse-like conditions might occur unintentionally when the width of a metal conductor along the direction of current flow is reduced.

The StarRC tool detects fuse conditions when the DETECT_FUSE command is set to YES. You must also specify the following commands:

- NETLIST_TAIL_COMMENTS: YES (to write the fuse information in the tail comments)
- REDUCTION: NO (to prevent the merging of fuse resistors)

Each detected fuse is represented in the netlist as a shorting resistor with a resistance value of 0.01 ohms, a length of 0, and a width. The reported width for abutting polygons is the length of the abutment region; the reported width of an overlap region is the length of the diagonal of the overlap region.

In the network, the fuse resistor is added to the wider of the two polygons because that is where the most narrowing along the direction of current flow occurs. You can report the fuse node location in the netlist by setting the EXTRA_GEOMETRY_INFO command to NODE and setting the NETLIST_NODE_SECTION command to YES.

Examples

Figure 190 and Figure 189 show fuses created from overlapping or abutting polygons. Figure 191 shows a tunnel pattern, in which a narrow polygon connects two wider polygons. L-shaped conductor configurations can be reduced to these cases. The rules also apply to versions of these patterns rotated by 90-degrees.

Feedback

In these figures, dimensions b1 and b2 represent the longer sides of the polygons; dimensions a1 and a2 are the shorter sides of the corresponding polygons.

The StarRC tool detects a fuse if dimension W is smaller than both the a1 and a2 dimensions. The fuse resistor width in the netlist is equal to dimension W.

Figure 189 Fuse Geometry With Abutting Polygons

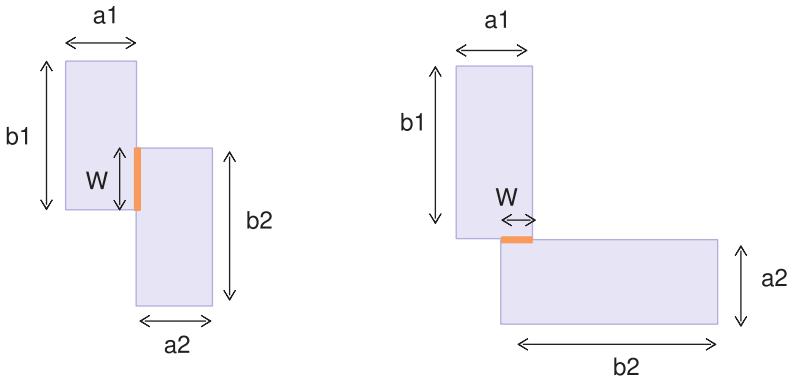
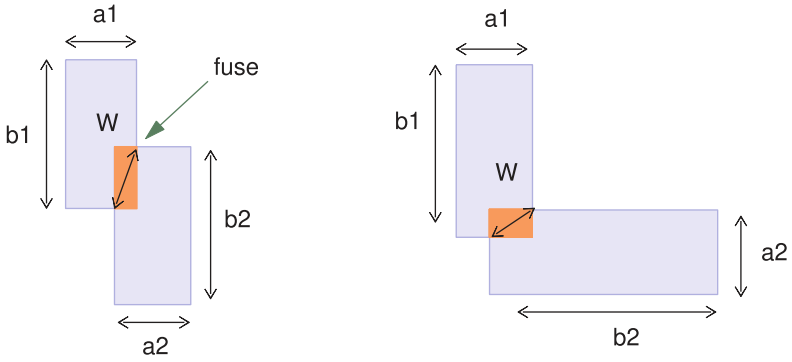
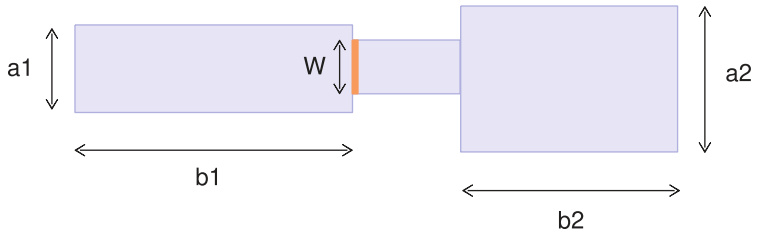


Figure 190 Fuse Geometry With Jog Patterns



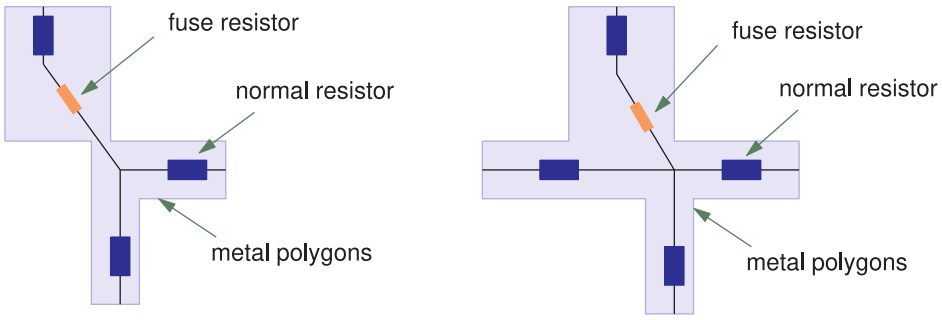
Feedback

Figure 191 Fuse Geometry With Tunnel Pattern



The fuse resistor location is shown in Figure 192. The greatest width narrowing occurs in the y-direction; therefore the fuse resistor is placed between the parasitic resistor for the widest net and the node where the metal polygons meet.

Figure 192 Fuse Resistor Location



An example netlist is as follows:

```
*|NET T1P2_12_050 0PF
*|P (T1P2_12_050 B 0 -229.71 160.863)//$llx=-230.705 $lly=160.858
$urx=-228.705 $ury=160.868 $lvl=4
*|P (T1P2_12_050_1 B 0 -229.71 170.847)//$llx=-231.205 $lly=170.847
$urx=-230.205 $ury=170.863 $lvl=4
*|S (67 -229.955 165.863)//$llx=-231.205 $lly=165.863 $urx=-228.705
$ury=165.863 $lvl=2
*|S (61 -229.955 165.863)//$llx=-230.705 $lly=165.863 $urx=-230.205
$ury=165.863 $lvl=2
*|S (F393 -230.71 170.847)//$llx=-231.205 $lly=170.847 $urx=-230.205
$ury=170.863 $lvl=4
*|S (F392 -229.71 160.863)//$llx=-230.705 $lly=160.858 $urx=-228.705
$ury=160.868 $lvl=4
RxT1P2_12_050_1 T1P2_12_050_1 F393 0.001
RxT1P2_12_050 T1P2_12_050 F392 0.001
R13 F392 67 1.647236 $l=5 $w=1 $lvl=2
```

Feedback

```
R14 67 61 0.01 $l=0 $w=0.5 $lvl=2
R15 61 F393 0.825000 $l=4.984 $w=2 $lvl=2
```

In this example. R14 is a fuse resistor with width 0.5. Subnode 61 is the fuse node. Its location is $llx=-230.705 $lly=165.863 $urx=-230.205 $ury=165.863 $lvl=2.

See Also

- NETLIST_TAIL_COMMENTS
- REDUCTION
- EXTRA_GEOMETRY_INFO
- NETLIST_NODE_SECTION

Feedback

DIELECTRIC_FILL_GDS_FILE

Specifies a GDSII file that contains dielectric fill data.

Syntax

```
DIELECTRIC_FILL_GDS_FILE: file_name
```

Arguments

Argument	Description
file_name	GDSII file containing dielectric fill data Default: none

Description

The DIELECTRIC_FILL_GDS_FILE and DIELECTRIC_FILL_GDS_LAYER_MAP_FILE commands in the StarRC command file define the dielectric fill shapes referenced by the DIELECTRIC_FILL_VS_SI_SPACING command in the ITF file. These three commands must be used together.

The StarRC tool treats as dielectric fill objects all shapes on layers in the mapping file (specified by the DIELECTRIC_FILL_GDS_LAYER_MAP_FILE command) within the master definition of the top block (specified by the BLOCK command) and its child cells. All data not referenced by the master definition of the top block is ignored.

The DIELECTRIC_FILL_GDS_FILE command supports both hierarchical and flat GDSII files. You can specify gzip compressed GDSII files.

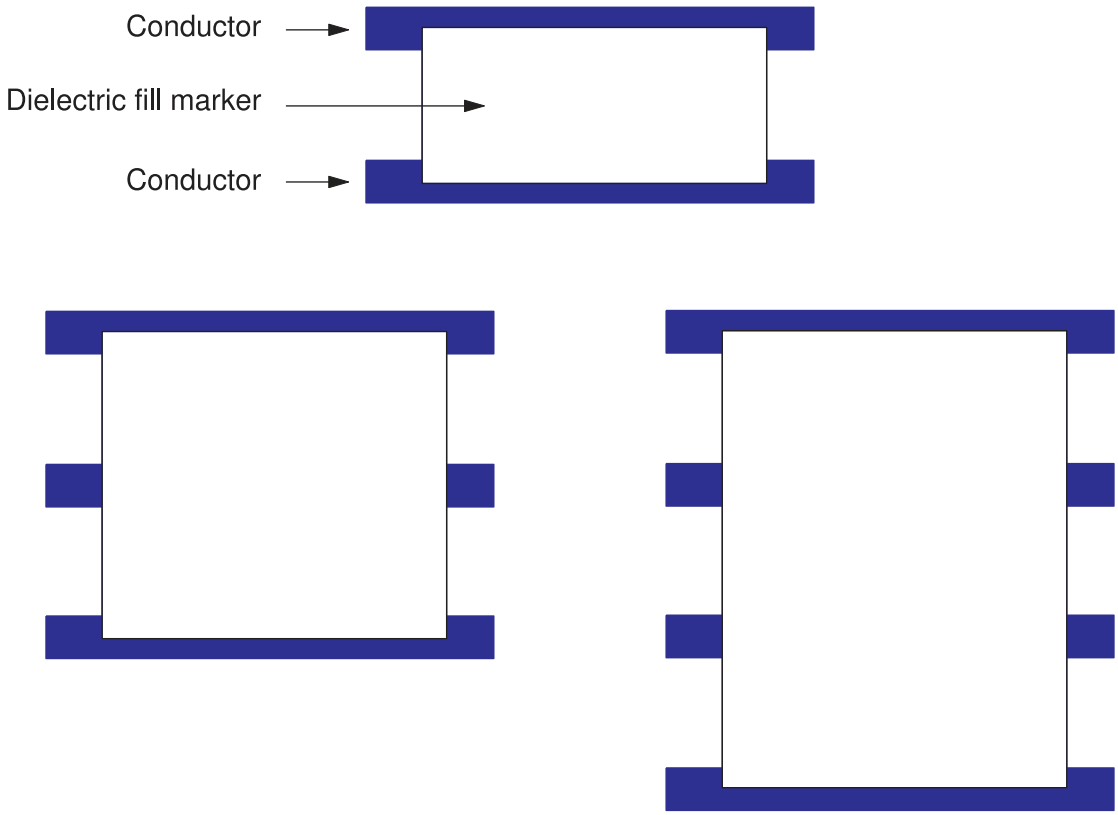
If you use more than one of the DIELECTRIC_FILL_GDS_FILE, METAL_FILL_GDS_FILE, and GDS_FILE commands in a single run, use a unified layer mapping file for the GDSII files.

Dielectric fill marker polygons must be defined from the center line of a conductor to the center line of another conductor on the same layer. One fill polygon can cover the spaces between multiple adjacent conductors on that layer.

The conductor shapes and the dielectric fill marker shapes should be placed on a metal database layer that is mapped to an ITF conductor layer that contains dielectric fill specified by the DIELECTRIC_FILL_VS_SI_SPACING option. Figure 193 shows examples of valid dielectric fill marker shapes.

Feedback

Figure 193 Examples of Dielectric Fill Marker Shapes



See Also

- DIELECTRIC_FILL_GDS_LAYER_MAP_FILE
- DIELECTRIC_FILL_VS_SI_SPACING
- GDS_FILE
- METAL_FILL_GDS_FILE
- The StarXtract -gdscheck Option

Feedback

DIELECTRIC_FILL_GDS_LAYER_MAP_FILE

Names the mapping file that specifies the GDSII layer numbers and layer names in the design database for dielectric fill features.

Syntax

```
DIELECTRIC_FILL_GDS_LAYER_MAP_FILE: file_name
```

Arguments

Argument	Description
file_name	Name of the GDSII layer mapping file Default: none

Description

The DIELECTRIC_FILL_GDS_FILE and DIELECTRIC_FILL_GDS_LAYER_MAP_FILE commands in the StarRC command file define the dielectric fill shapes referenced by the DIELECTRIC_FILL_VS_SI_SPACING command in the ITF file. These three commands must be used together.

The DIELECTRIC_FILL_GDS_LAYER_MAP_FILE command specifies the mapping file that maps the GDSII layer numbers and layer names in the design database whenever the DIELECTRIC_FILL_GDS_FILE command is used to import GDSII data into the design database. All translated GDSII layers must have an entry in the file specified by the DIELECTRIC_FILL_GDS_LAYER_MAP_FILE command and must have a definition in the layout database.

If you use more than one of the DIELECTRIC_FILL_GDS_FILE, METAL_FILL_GDS_FILE, and GDS_FILE commands in a single run, you can use a unified layer mapping file for the GDSII files.

Each line in the layer map file must use the following syntax:

```
database_layer gdsii_layer_number gdsii_datatype
```

Argument	Description
database_layer	The database layer name
gdsii_layer_number	The GDSII layer number
gdsii_datatype	The GDSII data type. If a GDSII data type is not specified, then all data types on a given layer are read.

Feedback

See Also

- DIELECTRIC_FILL_GDS_FILE
- DIELECTRIC_FILL_VS_SI_SPACING
- GDS_FILE
- METAL_FILL_GDS_FILE

Feedback

DIFFUSION_RES_MODE

Specifies the model to use for mesh source and drain diffusions.

Syntax

```
DIFFUSION_RES_MODE: LINE | POINT
```

Arguments

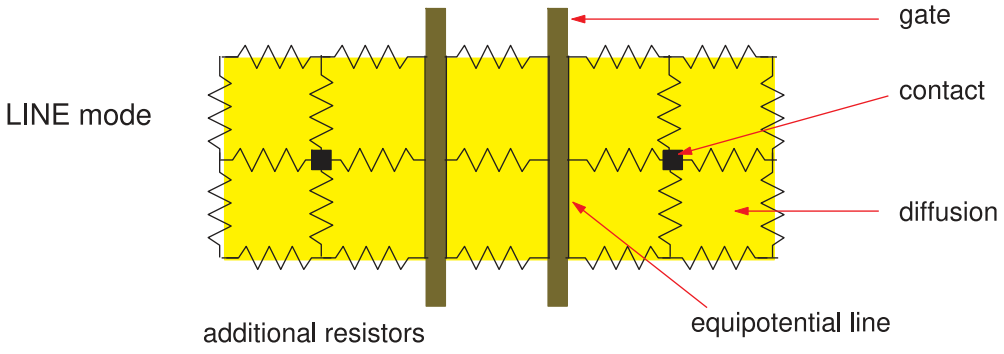
Argument	Description
`LINE` (default)	Models the gate to diffusion overlap region as an equipotential line
`POINT`	Models the gate to diffusion overlap region with resistors

Description

The StarRC tool extracts diffusion resistance as a resistor mesh structure. By default, the StarRC tool models the source and drain terminals of a transistor as an equipotential line, as shown in Figure 194.

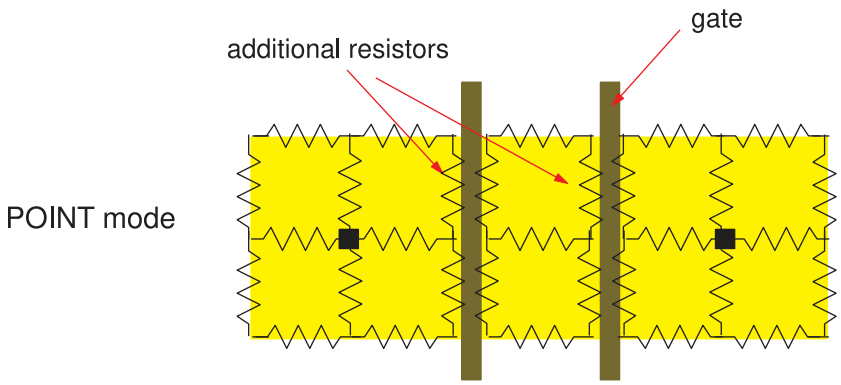
For advanced process nodes and transistors with large widths, the aspect ratio of the diffusion might be very large. The `POINT` mode inserts additional resistance along the source and drain contact area. Figure 194 and Figure 195 illustrate the two modes.

Figure 194 Diffusion Resistance LINE Mode



Feedback

Figure 195 Diffusion Resistance POINT mode



See Also

- Diffusion Resistance

Feedback

DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS

Treats connected pins inside skip cells as if they are not connected.

Syntax

```
DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS: YES | NO
```

Arguments

Argument	Description
YES	Allows top-level nets to be shorted inside skip cells
NO (default)	Does not short nets inside skip cells

Description

By default, the StarRC tool considers connected polygons to be part of the same net. If two nets in a cell are connected through lower-level cells, the tool merges the polygons of the two nets and assigns them to one of the nets. The missing net might cause parasitic annotation problems in downstream tools.

You can prevent this behavior by setting the DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS command to YES. In this case, if instance pins are connected inside a skip cell, they do not cause nets outside the cell to be shorted through those pins.

The DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS command interacts with the SHORT_PINS_IN_CELLS command as follows:

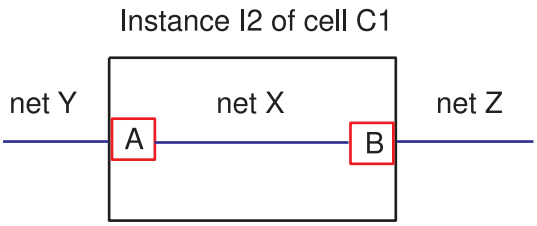
- If the SHORT_PINS_IN_CELLS command is set to !* or * !*cell_name*, the tool retains both nets connected to the skip cell pins from outside the skip cell.
- If the SHORT_PINS_IN_CELLS command is set to * or *cell_name*, the tool uses the net name as the pin name for that cell. In this case, the pin names are the same and they cannot be disconnected. Therefore the DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS command has no effect.

Examples

Figure 196 shows instance I2 of skip cell C1, which has two pins A and B. The pins are connected inside the cell by net X. From outside the cell, net Y connects to pin A (instance pin I2/A) and net Z connects to pin B (instance pin I2/B). By default, the StarRC tool recognizes that nets Y and Z are connected and assigns the polygons of net Z to net Y (or vice versa). In the output netlist, only one net (Y or Z) is reported.

Feedback

Figure 196 Connected Pins in a Skip Cell



Example 1. The StarRC command file contains these commands:

```
SHORT_PINS_IN_CELLS: !C1
DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS: NO
```

The output netlist is as follows. Note that the tool retains only one of net Y or net Z, but the specific net retained can be either net. The missing net might cause parasitic annotation issues in downstream tools.

```
*D_NET Y
*CONN
*I I2/A
*I I2/B
```

Example 2. The StarRC command file contains these commands:

```
SHORT_PINS_IN_CELLS: !C1
DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS: YES
```

The output netlist is as follows. Both nets Y and Z are present and connect to the appropriate pins of the cell.

```
*D_NET Y
*CONN
*I I2/A

*D_NET Z
*CONN
*I I2/B
```

Example 3. The StarRC command file contains these commands:

```
SHORT_PINS_IN_CELLS: C1
DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS: YES | NO
```

The output netlist is as follows. Both nets Y and Z are present, but the pins are named X for the net that connects them inside cell C1. In this case, the `DISCONNECT_PORT_AND_NET_IN_SKIP_CELLS` command has no effect.

```
*D_NET Y
*CONN
*I I2/X

*D_NET Z
*CONN
*I I2/X
```

See Also

- SHORT_PINS_IN_CELLS

Feedback

DPT

Enables extraction for double or multiple patterning processes.

Syntax

```
DPT: YES | NO
```

Arguments

Argument	Description
YES	Uses dielectric constant changes
NO (default)	Performs regular extraction

Description

Models the misalignment effects for multiple patterning technologies. If you set the DPT command to YES, the nxtgrd file must include either the ER_VS_SI_SPACING command or the LATERAL_CAP_SCALING_VS_SPACING command.

See Also

- ER_VS_SI_SPACING
- LATERAL_CAP_SCALING_VS_SPACING
- Double or Multiple Patterning Technology

Feedback

DPT_COLOR_GDS_FILE

Specifies the GDSII file containing the color-layer polygon definitions.

Syntax

```
DPT_COLOR_GDS_FILE: file_name
```

Arguments

Argument	Description
file_name	The GDSII file containing the color-layer polygon definitions

Description

The `DPT_COLOR_GDS_FILE` specifies the GDSII file containing the color-layer polygon definitions. This file is usually created by the place-and-route tool.

Examples

In the following example, the file named Decomposed_m1_m2.GDS contains the color-layer polygon definitions:

```
DPT_COLOR_GDS_FILE: Decomposed_m1_m2.GDS
```

See Also

- DPT
- DPT_COLOR_GDS_LAYER_MAP_FILE
- ER_VS_SI_SPACING
- SELECTED_CORNERS
- SIMULTANEOUS_MULTI_CORNER
- Double or Multiple Patterning Technology
- The StarXtract -gdscheck Option

Feedback

DPT_COLOR_GDS_LAYER_MAP_FILE

Maps color layers in the design database to GDSII layers.

Syntax

```
DPT_COLOR_GDS_LAYER_MAP_FILE: file_name
```

Arguments

Argument	Description
file_name	Mapping file for mapping color layers in the design database to the GDSII layers

Description

The `DPT_COLOR_GDS_LAYER_MAP_FILE` command specifies the file that maps the color layers in the design database to the GDSII layers. This file is created by the place-and-route tool and uses a pound sign (#) for comments.

An example of the file is as follows:

```
#DBLayerName        #GDS_LayerID      #GDS_LayerDataType
M1_color1               21                 1
M1_color2               21                 2
M2_color1               22                 1
M2_color2               22                 2
```

All layer names in the specified file must exist in the StarRC mapping file under the `color_layers` section.

Examples

In the following example, the `DPT_COLOR_GDS_LAYER_MAP_FILE` command specifies a file named gmap as the file that contains color layer mapping information.

```
DPT_COLOR_GDS_LAYER_MAP_FILE: gmap
```

See Also

- DPT
- DPT_COLOR_GDS_FILE
- Double or Multiple Patterning Technology

Feedback

ECO_MODE

Sets conditions for ECO extraction.

Syntax

```
ECO_MODE: YES | NO | RESET
```

Arguments

Argument	Description
YES	Specifies that ECO extraction should be performed if conditions are met
NO (default)	Specifies extraction of all nets with standard output (not suitable for ECO extraction)
RESET	Specifies extraction of all nets with output suitable for ECO extraction

Description

ECO (engineering change order) extraction is the technique of performing extraction only on parts of a design that are different from a reference design. This capability allows efficient evaluation of engineering change orders, especially when coupled with a timing analysis tool that can also operate on ECO netlists.

The ECO_MODE command controls ECO extraction. If you enable ECO extraction by setting the command to YES or RESET, you must also use the GPD command to specify the GPD directory name.

- The YES option allows the StarRC tool to perform the most appropriate extraction for the state of the ECO cycle. An ECO extraction is performed unless one of the following conditions applies:
  - If the design database does not have any logical or physical changes since the previous extraction, the StarRC tool does not perform any extraction and does not update the netlists.
  - If the ECO design changes are extensive and therefore the number of nets selected for re-extraction is large, the tool performs a full-chip extraction because there is little runtime benefit from an ECO extraction. This full-chip run becomes the reference for subsequent ECO runs.

- The RESET option causes the StarRC tool to perform a full-chip extraction in a manner compatible with subsequent ECO extractions. This option is a method to force a full-chip extraction during a cycle of ECO extractions.
- The NO option disables ECO extraction, which is equivalent to not using the command. In this case, the StarRC tool performs full-chip extraction.

The first StarRC run is always a full-chip extraction because a reference run must exist for ECO extractions. Subsequent StarRC runs might be either full-chip or ECO extractions, depending on the number of ECO-affected nets compared to the size of the design.

You cannot make changes in the command file that affect the extraction behavior between full-chip and ECO extractions or between successive ECO extractions. The commands can remain in the StarRC command file, but they cannot be modified between runs.

See Also

- GPD
- NETLIST_INCREMENTAL
- Chapter 5, ECO Extraction

Feedback

EM_PARAM_MAPPING_FILE

Specifies a file that describes parameters to be written to the parasitic netlist for use by third-party reliability analysis tools. Valid only for transistor-level flows.

Syntax

```
EM_PARAM_MAPPING_FILE: em_map_file
```

Arguments

Argument	Description
em_map_file	Electromigration (EM) parameter mapping file name Default: none

Description

The `EM_PARAM_MAPPING_FILE` command specifies the name of a mapping file that customizes a parasitic netlist for use by downstream electromigration (EM) and reliability analysis (RA) tools.

Note:

This command and the specified mapping file are for flows that use third-party EM and RA tools. Do not use this command with flows that use the Synopsys CustomSim tool.

You can specify which parameters to write to the netlist and provide custom labels. If you do not provide a mapping file, the netlist contains the default parameter names.

To use this feature, you must also specify the following commands:

- `NETLIST_TAIL_COMMENTS: YES`
- `EXTRA_GEOMETRY_INFO: NODE RES`
- `REDUCTION: NO`
- `KEEP_VIA_NODES: YES`
- `SHORT_PINS: NO`
- `NETLIST_NODE_SECTION: YES`
- `NETLIST_CONNECT_SECTION: YES`

Feedback

The precision of the reported geometric parameters is based on the precision of the input design files, as follows:

- If the input design precision is 1 nm, the number of digits after the decimal point is 3.
- If the input design precision is 0.1 nm or better (in other words, a smaller number), the number of digits after the decimal point is 4.
- For areas, the length and width values are multiplied and all digits after the decimal point are retained.

The non-physical resistors in SPF files can be identified based on categories by distinguishing the differences in resistance, width, and layer information in tail comments. The tool adds the `$NPHY_RES` tag to identify non-physical resistors in SPF files.

If both the `EM_EXTRACTION` and `NETLIST_TAIL_COMMENTS` commands are set to `YES`, an output SPF file contains the following information. In the following example, the `$NPHY_RES` tag helps you to identify R2_11 as a non-physical resistor.

```
R2_11 RR0@2:neg SUBOUT 0.001 $l=0.08922 $w=10.00000 $lvl=149 $NPHY_RES
```

EM Parameter Mapping File Syntax

The EM parameter mapping file uses the following syntax, where the two values are separated by one or more spaces. Begin a comment line with a pound sign (#).

```
# Comment line
PARAMETER_NAME     user_label
```

The first entry is one of the fixed parameter names. The second entry is a user-specified label to be written to the parasitic netlist for that parameter. Table 62 lists the available electromigration parameters.

The order of the parameters in the EM parameter mapping file does not matter. In the parasitic netlist, the order of the parameters that are reported for each device is fixed. However, some parameters might not be present, depending on the contents of the EM parameter mapping file.

Table 62 Parameters Available in Electromigration Parameter Mapping File

StarRC parameter name	Description
`RES_WIDTH`	Resistor width in microns
`RES_LENGTH`	Resistor length in microns
`RES_LAYER`	Resistor layer number, which corresponds to the layer number in the LAYER_MAP section of the netlist

Table 62 Parameters Available in Electromigration Parameter Mapping File (Continued)

StarRC parameter name	Description
RES_NONPHYSICAL_LAYER	Name for a new layer which contains all shorting resistors and which appears in the layer map section of the netlist
RES_VIA_AREA	Via area for a resistor on a via
RES_VIA_NUM	For a via array, the number of vias; for a trench contact virtual via, the reciprocal of the number of via segments
RES_MASK_ID	Mask ID of the layer if the resistor is on a contact layer with multiple masks
RES_CENTER_X	The x-coordinate of the center of the resistor
RES_CENTER_Y	The y-coordinate of the center of the resistor
RES_BBOX_LLX	The x-coordinate of the lower left corner of the resistor bounding box.
RES_BBOX_LLY	The y-coordinate of the lower left corner of the resistor bounding box.
RES_BBOX_URX	The x-coordinate of the upper right corner of the resistor bounding box.
RES_BBOX_URY	The y-coordinate of the upper right corner of the resistor bounding box.
SUPERCONDUCTIVE_VIA_LAYERS	Resistors on the specified database layers are identified as nonphysical (shorting) resistors; wildcards are allowed, for example, via* db_layer1 db_layer2.

Examples

Simple Resistor

Consider the following EM parameter mapping file:

```
RES_LENGTH      L
RES_WIDTH       W
RES_LAYER       lvl
RES_CENTER_X    X
RES_CENTER_Y    Y
```

This mapping file produces lines in a parasitic netlist similar to the following.

```
R3_1 VDD:1 VDD:2 1.05597 $m1 $L=5.3000 $W=1.0200 $lvl=13 $X=42.8550
$Y=811.200
```

All resistors display the ITF layer name (in the format $<*layer_name*>). The LAYER_MAP section of the netlist contains the ITF layer names.

Design With Via Arrays

The following EM parameter mapping file is for a design with via arrays:

```
RES_VIA_AREA    area
RES_VIA_NUM     num
RES_LENGTH      len
RES_WIDTH       wid
RES_LAYER       lvl
RES_CENTER_X    x
RES_CENTER_Y    y
```

This mapping file produces lines in a parasitic netlist similar to the following:

```
R4338_21 Xlat/Xrm_net50:1 Xlat/Xrm_net50:2 18.4 $VIA1 $area=0.002
$len=5.3000 $wid=1.0200 $lvl=13 $x=42.8550 $y=811.200 $num=6
```

Design With Virtual Vias

The following EM parameter mapping file is for a design with virtual vias:

```
RES_VIA_AREA    A
RES_VIA_NUM     N
RES_LENGTH      L
RES_WIDTH       W
RES_LAYER       LVL
RES_CENTER_X    X
RES_CENTER_Y    Y
```

This mapping file produces lines in a parasitic netlist similar to the following. The value of 0.500 for parameter $N is the reciprocal of the number of segments cut on the via, which in this case is 2.

```
R50011_90 Xclkb/X23/net11:1 Xclkb/X23/net11:2 270.209 $VIA2 $A=0.00045600
$L=0.0240 $W=0.0190 $LVL=22 $X=13.6230 $Y=2.9195 $N=0.5000
```

See Also

- NETLIST_TAIL_COMMENTS
- EXTRA_GEOMETRY_INFO
- REDUCTION
- KEEP_VIA_NODES
- SHORT_PINS
- NETLIST_NODE_SECTION

- NETLIST_CONNECT_SECTION
- Extraction For Electromigration Analysis

Feedback

ENABLE_IPV6

Specifies the network addressing protocol.

Syntax

```
ENABLE_IPV6: YES | NO
```

Arguments

Argument	Description
YES	Selects IPv6 for submit hosts that support both IPv4 and IPv6
NO	Selects IPv4 for submit hosts that support both IPv4 and IPv6
(default)	Detects and uses the correct mode (IPv4 or IPv6) for submit hosts that support only one mode. Do not use the ENABLE_IPV6 command for these hosts.

Description

The StarRC and grdgenxo tools can use either the IPv4 (32-bit) or IPv6 (128-bit) addressing protocol. The tool automatically detects the addressing mode on the submit host (the host used to launch jobs). The following usage notes apply:

- If the submit host supports both IPv4 and IPv6, you must set the ENABLE_IPV6 command to YES to use IPv6 or NO to use IPv4. If you omit the command and the submit host supports both address modes, the StarRC and grdgenxo tools issue an error message and stops.
- If the STARRC_DP_STRING command or environment variable specifies a list of machine names, the StarRC tool checks the mode of each host before submitting the jobs. If the GRD_DP_STRING command specifies a list of machine names, the grdgenxo tool checks the mode of each host before submitting the jobs. If any host is incompatible with the submit host, the tool issues an error message and stops.
- If the job is submitted to a compute farm, all hosts must support the address mode of the submit host. Remote jobs that land on an incompatible host fail.
- If the submit host supports only IPv4 or only IPv6, do not use the ENABLE_IPV6 command because the StarRC and grdgenxo tools detect the address mode.

This command does not apply to distributed processing jobs controlled by a GPD configuration file or by the FS_DP_STRING command.

Feedback

See Also

- STARRC_DP_STRING
- GRD_DP_STRING
- Distributed Processing
- Using Distributed Processing With the grdgenxo Tool

Feedback

ENHANCED_ALTERNATE_LAYER_COUPLING_CAP

Improves overlap capacitance extraction between shapes on alternate (next-to-next) layers in sparse situations. Valid only for gate-level flows.

Syntax

```
ENHANCED_ALTERNATE_LAYER_COUPLING_CAP: YES | NO
```

Arguments

Argument	Description
YES	Improves overlap capacitance extraction between shapes on alternate layers with respect to accuracy Default: NO
NO	Default

Description

In gate-level extraction, the command improves the extracted overlap capacitance accuracy, when the following conditions are met:

- The shapes on alternate layers overlap without the shapes present in the middle layer
- In combination with the sparse layout conditions

By default, the command is set to ON for transistor-level flow.

This command does not apply to field solver extractions specified with the FS_EXTRACT_NETS command.

Feedback

ENHANCED_GPD_POWER_REDUCTION

Enables a targeted reduction operation for power nets. Valid only for transistor-level GPD flows.

Syntax

```
ENHANCED_GPD_POWER_REDUCTION: YES | NO
```

Arguments

Argument	Description
YES	Enables enhanced power net reduction
NO (default)	Disables enhanced power net reduction

Description

Power nets present a challenge for transistor-level extraction. The number of power nets is often very large and the accuracy requirements are not as stringent as for signal nets. The enhanced reduction method reduces memory usage and the number of parasitic devices on the power net, which results in reduced runtime for downstream simulation. The greatest improvement occurs for designs with large power grids.

Set the ENHANCED_GPD_POWER_REDUCTION command to YES to enable a specialized reduction algorithm for power nets. The requirements are as follows:

- The power nets must be identified implicitly in the design database or in a POWER_NETS command in the StarRC command file.
- You must set the POWER_EXTRACT command to YES.
- You must run a valid GPD flow. If the command file contains a command that is not supported for transistor-level GPD creation, the tool reverts to a standard extraction flow and enhanced power net reduction does not occur.

Setting the ENHANCED_GPD_POWER_REDUCTION command to YES automatically sets the POWER_REDUCTION command to HIGH.

See Also

- POWER_NETS
- POWER_REDUCTION
- REDUCTION

Feedback

ENHANCED_SHORT_REPORTING

Specifies whether to generate an enhanced shorts report. Valid only for gate-level flows.

Syntax

```
ENHANCED_SHORT_REPORTING: YES | NO | COMPLETE
```

Arguments

Argument	Description
NO (default)	Generates a standard shorts report
YES	Generates an enhanced shorts report
COMPLETE	Generates an enhanced shorts report that also includes shorts to internal nets of skip cells

Description

The ENHANCED_SHORT_REPORTING command specifies the types of shorts to be included in the shorts report: standard or enhanced. The StarRC tool evaluates potential shorts and reports a subset of them in a file named *shorts_all.sum* located in the ./star directory.

The StarRC tool identifies the following specified nets as skip cell material:

- The non-critical net specified with the COUPLE_NONCRITICAL_NETS command
- The skip cell net specified with the SKIP_CELLS_COUPLE_TO_NET command

Therefore, the shorts to them are reported as shorts to skip cells (category #20 in the Table 63) only when ENHANCED_SHORT_REPORTING: COMPLETE.

Table 63 lists the shorts that are reported in the *shorts_all.sum* file for the settings of the ENHANCED_SHORT_REPORTING command.

Table 63 Shorts Reported With ENHANCED_SHORT_REPORTING

Category	First polygon	Second polygon	NO	YES	COMPLETE	Second polygons	Short Report Format in shorts_all.sum
1	Extracted signal net	Extracted signal net	yes	yes	yes	Extracted signal net	Short between <net1> [of instance <> of cell <>] to <net2> [of instance <> of cell <>]

Feedback

Table 63 Shorts Reported With ENHANCED_SHORT_REPORTING (Continued)

Category	First polygon	Second polygon	NO	YES	COMPLETE	Second polygons	Short Report Format in shorts_all.sum
2	Extracted signal net	Port of extracted net	yes	yes	yes	Top-level ports of extracted net	Short between <net1> [of instance <> of cell <>] to <net2>
3	Extracted signal net	Skip cell ports of extracted nets	yes	yes	yes	SKIP_CELLS ports of extracted nets	Short between <net1> [of instance <> of cell <>] to <net2> of instance <> of cell <>
4	Port of extracted net	Port of extracted net	yes	yes	yes		Short between <net1> to <net2>
5	Port of extracted net	Skip cell ports of extracted nets	yes	yes	yes		Short between <net1> to <net2> of instance <> of cell <>
6	Extracted signal net	Floating fill	yes	yes	yes	Floating metal fill	Short between <net1> [of instance <> of cell <>] to floating fill
7	Port of extracted net	Floating fill	yes	yes	yes	Floating metal fill	Short between <net1> to floating fill
8	Extracted signal net	Grounded fill	yes	yes	yes	Grounded metal fill	Short between <net1> [of instance <> of cell <>] to grounded fill
9	Port of extracted net	Grounded fill	yes	yes	yes	Grounded metal fill	Short between <net1> to grounded fill
10	Extracted signal net	Grounded SADP fill	yes	yes	yes	SAPD metal fill	Short between <net1> [of instance <> of cell <>] to SADP fill
11	Port of extracted net	Grounded SADP fill	yes	yes	yes	SAPD metal fill	Short between <net1> to SADP fill
12	Extracted signal net	Unselectable signal net	no	yes	yes	Unselectable signal net	Short between <net1> [of instance <> of cell <>] to unselectable net [of instance <> of cell <>]

Feedback

Table 63 Shorts Reported With ENHANCED_SHORT_REPORTING (Continued)

Category	First polygon	Second polygon	NO	YES	COMPLETE	Second polygons	Short Report Format in shorts_all.sum
13	Port of extracted net	Unselectable signal net	no	yes	yes	Unselectable signal net	Short between <net1> to unselectable net [of instance <> of cell <>]
14	Extracted signal net	Non-extracted power net	no	yes	yes	Power nets when POWER_EXTRACT_NO	Short between <net1> [of instance <> of cell <>] to power net <net name> [of instance <> of cell <>]
15	Port of extracted net	Non-extracted power net	no	yes	yes	Power nets when POWER_EXTRACT_NO	Short between <net1> to power net <net name> [of instance <> of cell <>]
16	Extracted signal net	Non-extracted power net	no	yes	yes	Unselected signal nets	Short between <net1> [of instance <> of cell <>] to nonselected net
17	Port of extracted net	Non-extracted signal net	no	yes	yes	Unselected signal nets	Short between <net1> to nonselected net
18	Extracted signal net	Skip cell blockage polygon	no	yes	yes	Polygons marked as blockage from LEF, or Milkyway or NDM FRAME view	Short between <net1> [of instance <> of cell <>] to blockage of instance <> of cell <>
19	Port of extracted net	Skip cell blockage polygon	no	yes	yes	Polygons marked as blockage from LEF, or Milkyway or NDM FRAME view	Short between <net1> to blockage of instance <> of cell <>
20	Extracted signal net	Skip cell blockage polygon	no	no	yes	Hierarchical and internal polygons inside SKIP_CELLS, which might be from MACRO_DEF_FILE, MILKYWAY_CELL_VIEW and NDM_DESIGN_VIEW, or GDS_FILE/OASIS_FILE	Short between <net1> [of instance <> of cell <>] to skip cell of instance <> of cell <>

Feedback

Table 63 Shorts Reported With ENHANCED_SHORT_REPORTING (Continued)

Category	First polygon	Second polygon	NO	YES	COMPLETE	Second polygons	Short Report Format in shorts_all.sum
	Port of extracted net	Skip cell polygon	no	no	no		none

When the POWER_EXTRACT command is set to NO and the ENHANCED_SHORT_REPORTING command is set to YES or COMPLETE, the StarRC tool reports shorts in the *shorts_all.sum* file from the *extracted signal nets to a non-extracted power net*, as follows:

```
Short between net A and power net vss Layer=M6 Bbox=(447.052,436.477), \
(447.097,436.477)
```

Examples

Shorts in the *shorts_all.sum* file are reported in the following format:

```
Short between net vss and net UNSIG_295 Layer=M6 Bbox=(444.204,427.342),
 (444.204,427.387)

Short between net vss and unselected net Layer=M6 Bbox=(447.052,436.477),
 (447.097,436.477)

Short between net vss and blockage Layer=M6 Bbox=(445.342,444.082),
 (445.387,444.127)
```

The coordinates for the bounding boxes are in units of microns. Coordinates might be scaled or unscaled depending on the settings of the HALF_NODE_SCALE_FACTOR, MAGNIFICATION_FACTOR, and NETLIST_UNSCALED_COORDINATES commands.

See Also

- MAGNIFICATION_FACTOR
- NETLIST_UNSCALED_COORDINATES
- HALF_NODE_SCALE_FACTOR
- Shorts Reports

Feedback

ESTIMATED_CONNECT_OPEN_NET

Generates only for the specified nets a complete netlist by identifying open nets and inserts appropriate estimated RC parasitics for the missing net segments.

Syntax

```
ESTIMATED_CONNECT_OPEN_NET: <netname> <est_horiz_route_layer>
                  <est_vert_route_layer> <small_gap_threshold>
```

Arguments

Argument	Description
`netname`	Name of the net.
`est_horiz_route_layer`	Database layer for estimated horizontal routes.
`est_vert_route_layer`	Database layer for estimated vertical routes.
`small_gap_threshold`	Distance in microns. When the distance between nodes to be connected is more than the `small_gap_threshold` value, estimated routes that connect the nodes are placed on user-specified layers.

Description

Before you use the `ESTIMATED_CONNECT_OPEN_NET` command, you must first specify the `ESTIMATED_CONNECT_OPENS` command. Otherwise, the tool issues an error message.

For more information about using the `ESTIMATED_CONNECT_OPEN_NET` command, see ESTIMATED_CONNECT_OPENS.

See Also

- ESTIMATED_CONNECT_OPENS
- ESTIMATED_CONNECT_OPENS_VIAR_SCALE
- ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET

Feedback

ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET

Scales all via resistance values only for the specified net with the specified value.

Syntax

```
ESTIMATED_CONNECT_OPENS_VIAR_SCALE: <netname> <value>
```

Arguments

Argument	Description
`netname`	Name of the net.
`value`	Scaling factor to scale via resistance values A positive, nonzero scale factor

Description

The `ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET` command scales via resistance values that are inserted by the `ESTIMATED_CONNECT_OPENS` command. You can specify the command multiple times in the StarRC command file.

Note:

Before you use the `ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET` command, you must first specify the `ESTIMATED_CONNECT_OPENS_VIAR_SCALE` command. Otherwise, the tool issues an error message.

For more information about using the `ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET` command, see ESTIMATED_CONNECT_OPENS_VIAR_SCALE.

See Also

- ESTIMATED_CONNECT_OPENS_VIAR_SCALE
- ESTIMATED_CONNECT_OPENS
- ESTIMATED_CONNECT_OPEN_NET

Feedback

ESTIMATED_CONNECT_OPENS

Generates a complete netlist by identifying all open nets and inserts appropriate estimated RC parasitics for the missing net segments.

Syntax

```
ESTIMATED_CONNECT_OPENS: <est_horiz_route_layer> <est_vert_route_layer>
                         <small_gap_threshold>
```

Arguments

Argument	Description
`est_horiz_route_layer`	Database layer for estimated horizontal routes.
`est_vert_route_layer`	Database layer for estimated vertical routes.
`small_gap_threshold`	Distance in microns. When the distance between nodes to be connected is more than the `small_gap_threshold` value, estimated routes that connect the nodes are placed on user-specified layers.

Description

You must first specify the `ESTIMATED_CONNECT_OPENS` command in the StarRC command file before you use the `ESTIMATED_CONNECT_OPEN_NET` command. Otherwise, the tool issues an error message.

The routes to complete open nets are estimated based on the following criteria:

- User-specified metal layers:
  - Horizontal (Mx) and vertical (My) routes specified with the `est_horiz_route_layer` and `est_vert_route_layer` arguments respectively
  - Appropriate vias are selected between layers
- The estimated routes are placed on the same layer where the nodes are connected (instead of using Mx and My layers) when the distance between the nodes is less than the `small_gap_threshold` value.
- Nodes on the highest metal layer (closest to the user-specified metal layers) are selected from each resistively connected group for connecting open nets when connecting the resistively connected groups.

Feedback

Parasitic resistances and capacitances are calculated for the estimated routes based on the following criteria:

- Resistance for the estimated routes is calculated based on the minimum width for the corresponding metal layer.
- Capacitance for the estimated routes is calculated based on a pessimistic capacitance model, assuming minimum spacing from neighbors on the same layer and the perpendicular routes on neighboring layers.

Examples

The following example shows how the `ESTIMATED_CONNECT_OPEN*` commands are specified in the StarRC command file:

```
ESTIMATED_CONNECT_OPENS: M3 M4 1.0
ESTIMATED_CONNECT_OPEN_NET: clk_net M7 M8 0.5
```

See Also

- ESTIMATED_CONNECT_OPEN_NET
- ESTIMATED_CONNECT_OPENS_VIAR_SCALE
- ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET

Feedback

ESTIMATED_CONNECT_OPENS_VIAR_SCALE

Scales all via resistance values based on a specified value.

Syntax

```
ESTIMATED_CONNECT_OPENS_VIAR_SCALE: <value>
```

Arguments

Argument	Description
value	Scaling factor to scale via resistance values A positive, nonzero scale factor

Description

The ESTIMATED_CONNECT_OPENS_VIAR_SCALE command scales via resistance values that are inserted by the ESTIMATED_CONNECT_OPENS command.

Note:

You must first specify the ESTIMATED_CONNECT_OPENS_VIAR_SCALE command before you use the ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET command.

Examples

The following example shows scaling of via resistance values:

```
ESTIMATED_CONNECT_OPENS_VIAR_SCALE: 1
ESTIMATED_CONNECT_OPENS_VIAR_SCALE_NET: x_nt 0.5
ESTIMATED_CONNECT_OPENS_VIAR_SCALE_NET: y_nt 0.75
```

Consider a via from layer A to layer B with the via resistance value of 2. The x_nt, y_nt, and z_nt nets have an open that are connected from layer A to layer B. Then, the nets have the following via resistance value:

- x_nt with the via resistance value of 1
- y_nt with the via resistance value of 1.5
- z_nt with the via resistance value of 2

The specified x_nt and y_nt nets are scaled using the ESTIMATED_CONNECT_OPENS_VIAR_SCALE and ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET commands.

Feedback

See Also

- ESTIMATED_CONNECT_OPENS
- ESTIMATED_CONNECT_OPEN_VIAR_SCALE_NET
- ESTIMATED_CONNECT_OPEN_NET

Feedback

EVACCESS_DIRECTORY

Specifies the location of the Hercules LVS EvAccess database.

Syntax

```
EVACCESS_DIRECTORY: path
```

Arguments

Argument	Description
path	Path to the Hercules LVS EvAccess database Default: none

Description

Specifies the location of the Hercules LVS EvAccess database. This path can also be specified in the Hercules runset `EVACCESS_OPTIONS` section, with the `PATH` option. The Hercules default for this option is `./evaccess`.

If this path is not specified and the `XREF` command is set to `YES` or `COMPLETE`, the tool attempts to read the directory location from the XTR (Milkyway Extract) view.

The `EVACCESS_DIRECTORY` command is not used in the IC Validator flow.

See Also

- XREF

Feedback

EXCLUDE_STDCELLS_FROM_SHORT_PINS_IN_CELLS

Specifies if standard cells should be excluded from the `SHORT_PINS_IN_CELLS` command report.

Syntax

```
EXCLUDE_STDCELLS_FROM_SHORT_PINS_IN_CELLS | NO | YES
```

Arguments

Argument	Description
`No` (default)	Outputs netlist with all terminals for port, when the `SHORT_PIN_IN_CELLS` command is set to `!*`
`YES`	Outputs only one terminal for port, irrespective of the `SHORT_PIN_IN_CELLS` command setting

Description

Some standard cells (for example, clock tree buffers) can have multiple terminals. But, the .lib/synopsys.db database used in the PrimeTime tool does not have multiple terminals for the standard cells. However, the StarRC tool can read the Fusion Compiler or IC Compiler Mikyway and NDM designs to identify the standard cells and exclude them in the multiple physical-pins hierarchical flow. For more information, see Multiple Physical-Pins Hierarchical Flow.

The StarRC tool checks whether the standard cell has the `design_type` attribute and if the attribute is equal to `LIB_CELL`, the tool considers it as a standard cell. All electrically equivalent pins of the standard cells are merged into a single port. When the `EXCLUDE_STDCELLS_FROM_SHORT_PINS_IN_CELLS` command is set to `YES` in the multiple physical-pins hierarchical flow, the command outputs only one terminal for the ports of the standard cells that are set with the `SHORT_PIN_IN_CELLS: !*` command.

See Also

- MULTI_PHYSICAL_PINS_PREFIX
- SHORT_PINS_IN_CELLS

Feedback

EXPLODE_TRIVIAL_INSTANCE_PORTS

Explodes skip-cell (black-box cell) ports that are not connected to devices.

Syntax

```
EXPLODE_TRIVIAL_INSTANCE_PORTS: NO | YES
```

Arguments

Argument	Description
NO (default)	Does not explode trivial-instance ports
YES	Explodes trivial-instance ports

Description

By definition, a trivial port is not connected to any devices, considering the complete hierarchy, and is untexted in the layout.

Trivial ports are sometimes created by a layout versus schematic tool when noncritical material (such as metal fill) overlaps both a skip cell and its parent cell.

Trivial ports are labeled with a prefix set by the XREF_LAYOUT_NET_PREFIX command. The default prefix is In_. These ports do not exist in the schematic or Verilog and their existence might cause back-annotation issues in downstream tools, such as timing analysis tools.

When you set the EXPLODE_TRIVIAL_INSTANCE_PORTS command to YES, the StarRC tool removes trivial-instance ports from the netlist by exploding them to the parent level. When the trivial-instance port is removed, the polygons of the port are exploded to become a part of the top-level net. The resistance and capacitance are extracted on the polygons based on the EXTRACTION command setting. The REDUCTION command setting also might further affect the RC network.

In addition, setting this command to YES removes trivial ports in a SPICE file specified by the NETLIST_IDEAL_SPICE_FILE command. However, if a SPICE file specified by the SPICE_SUBCKT_FILE contains port definitions for skip cells, those ports are always retained.

See Also

- NETLIST_IDEAL_SPICE_FILE
- REMOVE_TRIVIAL_INSTANCE_PORTS
- DELETE_TRIVIAL_INSTANCE_PORTS

- SPICE_SUBCKT_FILE
- XREF_LAYOUT_NET_PREFIX

Feedback

EXTRA_GEOMETRY_INFO

Reports the internal node bounding box information for resistors.

Syntax

```
EXTRA_GEOMETRY_INFO: RES | NODE | RES NODE | NODE RES | NONE
```

Arguments

Argument	Description
RES	Reports bounding boxes for metal and via resistors
NODE	Reports bounding boxes for nodes
RES NODE	Reports both resistor and node information
NODE RES	Identical to RES NODE
NONE (default)	Does not report extra geometry information

Description

The EXTRA_GEOMETRY_INFO command reports the internal node bounding box information for a resistor, either as a tail comment in the node section of the netlist, a node property in the Milkyway parasitic database, or both. The bounding box dimensions are always as drawn and are not affected by the NETLIST_UNSCALED_COORDINATES command.

If the EXTRA_GEOMETRY_INFO command appears more than one time in a StarRC command file, only the last setting is used.

The following notes apply to the RES setting:

- You must set the NETLIST_TAIL_COMMENTS command to YES to write the extra resistor information to the netlist.
- You must set the REDUCTION and POWER_REDUCTION commands to NO to preserve the original layout topology.
- If you want to preserve via resistors, set the KEEP_VIA_NODES command to YES.

The following notes apply to the NODE setting:

- All reduction modes are supported
- If you want to preserve via resistors, set the KEEP_VIA_NODES command to YES.

If you are performing power rail analysis, set the TARGET_PWRA command to YES. This command automatically sets StarRC commands for optimal analysis, including setting the EXTRA_GEOMETRY_INFO command to RES NODE.

Bounding boxes are reported with the following four coordinates in the tail comments:

- $llx is the lower-left x-coordinate (all coordinates are in microns)
- $lly is the lower-left y-coordinate
- $urx is the upper-right x-coordinate
- $ury is the upper-right y-coordinate

In addition, the direction of current flow is reported with the $dir label: 0 for horizontal, 1 for vertical, and 2 for non-Manhattan.

Examples

This example shows part of a netlist with the EXTRA_GEOMETRY_INFO: NODE RES setting:

```
*D_NET I20|N9 0.0869015

*CONN
*I I20|I44.X O *C 151.19 11.75 *D NAN2 $llx=150.35 $lly=11.75 \
$urx=151.19 $ury=12.73 $lvl=1
*I I20|I45.A I *C 149.16 11.75 *L 2 *D NOR2 $llx=149.16 $lly=11.75 \
$urx=149.16 $ury=12.73 $lvl=1
*N I20|N9.220 *C 155.04 17.42 // $llx=154.83 $lly=17.42 $urx=155.04 \
$ury=17.42 $lvl=1
*N I20|N9.235 *C 154.83 18.68 // $llx=154.83 $lly=18.68 $urx=155.04 \
$ury=18.68 $lvl=1
*N I20|N9.234 *C 153.57 18.68 // $llx=153.57 $lly=18.68 $urx=153.57 \
$ury=18.68 $lvl=1

1: *84.1718 *84.1705 0.646554 // $l=0.152000 $w=0.114000 $lvl=4 \
$llx=102.490 $lly=64.576 $urx=102.605 $ury=64.823 $dir=0
```

When you run an extraction using EXTRA_GEOMETRY_INFO, the LAYER_MAP section of the netlist can also contain generated layer names. Extra layers are formed in the case of device-level extraction when there are database layers at the diffusion level or below that share a contact. For instance, if the runset contains the line shown in the following example, then the LAYER_MAP section contains an extra layer called nsd:psd or psd:nsd, which becomes the lower terminal level of diffCont via resistors.

```
CONNECT metal1 nsd psd BY diffCont
```

See Also

- CAPACITOR_TAIL_COMMENTS
- NETLIST_TAIL_COMMENTS

Feedback

EXTRACT_RES_BODY_COUPLING

Specifies the extraction of coupling capacitances between resistor body elements and interconnect features or ground.

Syntax

```
EXTRACT_RES_BODY_COUPLING: YES | NO
```

Arguments

Argument	Description
YES	Enables resistor body capacitance extraction
NO (default)	Disables resistor body capacitance extraction

Description

The EXTRACT_RES_BODY_COUPLING command specifies the extraction of coupling capacitances between resistor body elements and interconnect features or ground. The coupling capacitance between a resistor body and interconnect layers is distributed between the two terminals of the resistor.

This command does not affect the treatment of other coupling capacitances that are on nets connected to the resistor terminals (in other words, coupling capacitances that are not associated with the resistor body element).

If you set the EXTRACT_RES_BODY_COUPLING command to YES, resistor body capacitances are retained during extraction regardless of the settings of the COUPLE_TO_GROUND and RETAIN_GATE_CONTACT_COUPLING commands.

If you set the NETLIST_TYPE command to a setting other than CC or RCC, the resistor body capacitances are grounded.

This command is valid for IC Validator, Hercules, and Calibre flows.

See Also

- COUPLE_TO_GROUND
- NETLIST_TYPE
- RETAIN_GATE_CONTACT_COUPLING

Feedback

EXTRACT_RES_BODY_RESISTANCE

Specifies the names of design resistors to extract resistance.

Syntax

```
EXTRACT_RES_BODY_RESISTANCE: resistor_models
```

Arguments

Argument	Description
resistor _models	List of resistor models for which body resistance is extracted

Description

You can use the `EXTRACT_RES_BODY_RESISTANCE` command to extract the resistance of the specified design resistors that are not included in the simulation models. Then, the StarRC tool splits the extracted resistance value into two halves and adds them at each terminal of the design resistor, as explained in the following steps:

1. The command extracts the resistance of the specified design resistor.
2. The resistor for the specified model is represented as two resistors.

   Each resistor is specified with *l = 0* and *Resistance = BR/2* (body resistance divided by 2). The two split resistors are connected to each of the terminals of the specified design resistor. You can enable this only if the resistance is not accounted in the simulation models.

Figure 197 illustrates how the `EXTRACT_RES_BODY_COUPLING` and `EXTRACT_RES_BODY_RESISTANCe` commands extract coupling capacitance and resistance of the design resistor. In Figure 197,

- DR1: Indicates the design resistor. X and Y are the terminals of the design resistor.

  The body is shown in yellow.

- DR2: Indicates that the StarRC tool extracts the resistor body coupling capacitance and resistance. The StarRC tool calculates the resistance using the following formula:

  $$R = \mathrm{Rhosi} \times \frac{\mathrm{Lsi}}{(\mathrm{Tsi} \times \mathrm{Wsi})}$$

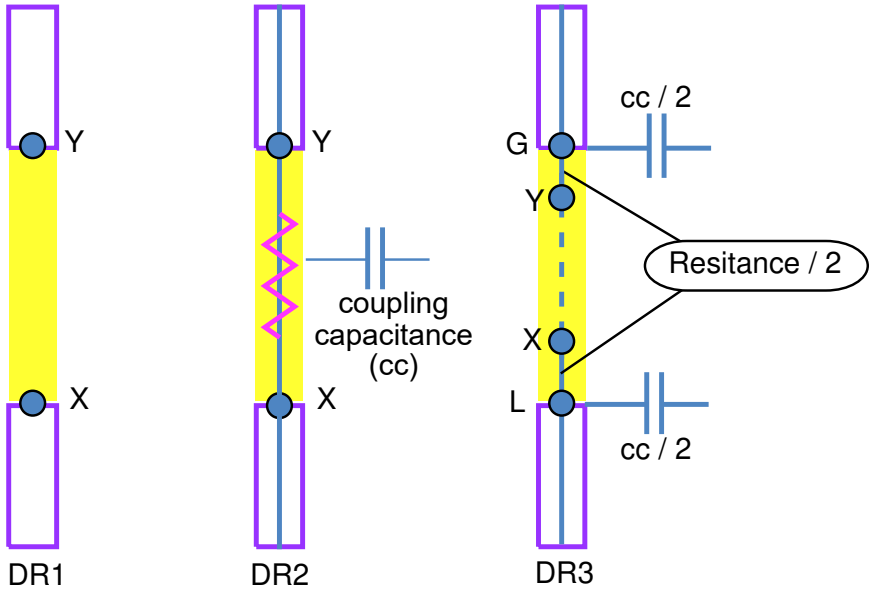
  Where, RhoSi is resistivity, Lsi is length, Tsi is thickness, and Wsi is width of the resistor body.

- DR3: Shows how the coupling capacitance and resistance of the resistor body is distributed. The dotted line indicates that the resistance of a device model is set to 0.001 (`R=0.001`).

  Note:

  Y and G terminals are exactly at the same location, and X and L terminals are exactly at the same location. In Figure 197 (DR3), they are shifted to show there presence.

Figure 197 Extracting body resistance and coupling capacitance



See Also

- COUPLE_TO_GROUND
- NETLIST_TYPE
- RETAIN_GATE_CONTACT_COUPLING

Feedback

EXTRACT_SPECIFIED_PCELL_PINS

Specifies the resistance extraction mode to pins of specific parameterized cell (PCell). Valid only for transistor-level flows.

Syntax

```
EXTRACT_SPECIFIED_PCELL_PINS: SUPERCONDUCTIVE | CONDUCTIVE cell_name
```

Arguments

Argument	Description
SUPERCONDUCTIVE (default)	Extracts PCell pins with zero resistivity
CONDUCTIVE	Extracts PCell pins with normal resistivity
cell_name	The cell to which the command applies. Wildcards * and ? are accepted

Description

The EXTRACT_SPECIFIED_PCELL_PINS command is a cell-specific command that controls whether the StarRC tool extracts PCell pin with resistivity. This command requires a setting and a cell name as arguments.

The following usage notes apply:

- You can use the * and ? wildcards in the cell name. The cell name is case-sensitive if the CASE_SENSITIVE command is set to YES (the default).
- You can use this command multiple times in a StarRC command file. If specific PCells are named multiple times, later instances of this command overwrite earlier instances.
- All cells named in this command must also be named in a SKIP_PCELLS command. If a cell name does not appear in both commands, the tool issues a warning message and ignores the EXTRACT_SPECIFIED_PCELL_PINS command setting for that cell.

Examples

The following example extracts resistance from the ports of the my_pcell* PCells:

```
EXTRACT_SPECIFIED_PCELL_PINS: CONDUCTIVE my_pcell*
```

The following example extracts only nonphysical resistors from ports of PCells when you set the EXTRACT_SPECIFIED_PCELL_PINS command to SUPERCONDUCTIVE (the default):

```
EXTRACT_SPECIFIED_PCELL_PINS: SUPERCONDUCTIVE
   R1_123 net:1 net:2 0.001 $lvl=180 $X=0 $Y=0
```

Feedback

The StarRC tool adds the `$pcell_res=1` flag in a netlist file as shown in the following example, so the electromigration (EM) tool skips performing the EM check on the specified resistor when the `EXTRACT_SPECIFIED_PCELL_PINS` command is set to `CONDUCTIVE` and the `REDUCTION: NO` and `NETLIST_TAIL_COMMENTS: YES` commands are also specified in the StarRC command file.

```
EXTRACT_SPECIFIED_PCELL_PINS: CONDUCTIVE
   R1_123 net:1 net:2 50 $lvl=180 $X=0 $Y=0 $pcell_res=1
```

See Also

- REDUCTION
- NETLIST_TAIL_COMMENTS
- PCELL_EXTRACTION_FILE

Feedback

EXTRACT_VIA_CAPS

Performs a detailed via capacitance extraction.

Syntax

```
EXTRACT_VIA_CAPS: NO | YES [IGNORE_GATE_CONTACT_COUPLING]
```

Arguments

Argument	Description
NO (default)	Ignores capacitive effect of vias and contacts. In general, this setting uses less runtime and reports fewer capacitances. Best used while developing designs and for technology nodes of 130 nm and above.
YES	Considers the capacitive effect of vias and contacts. Provides best accuracy at the cost of greater runtime. Best used for signoff designs and those with technology nodes of 90 nm and below.
YES IGNORE_GATE_CONTACT_COUPLING	Ignores the gate contact coupling if SPICE models include gate-to-contact capacitance.

Description

This command is used in conjunction with the EXTRACTION command.

If the EXTRACT_VIA_CAPS command is set to YES, the StarRC tool considers the capacitive effects of all via and contact layers.

For best accuracy, specify the GATE_TO_CONTACT_SMIN option in the ITF file in addition to the SMIN option inside the ITF CONDUCTOR definition for polysilicon gate layers. This allows the StarRC tool to use the actual gate-to-contact spacing when extracting contact capacitance; this spacing is typically smaller than the standard polysilicon SMIN spacing.

See Also

- EXTRACTION

Feedback

EXTRACTION

Specifies the type of extraction and the scope of the generated netlist.

Syntax

```
EXTRACTION: RC | C | R | FSCOMPARE | NORC
```

Arguments

Argument	Description
RC (default)	Extracts both parasitic resistor and capacitor devices and merges them into the original database network to produce a consolidated RC network description of the layout in the specified format.
C	Extracts only parasitic capacitor devices and produces a merged parasitic layout network description as a SPICE file. The NETLIST_FORMAT command is ignored for capacitance-only extractions.
R	Extracts only parasitic resistor devices and produces a merged parasitic layout network description in the specified format.
FSCOMPARE	Provides a comparison report of a merged layout network description containing only parasitic capacitors, executes a field solver analysis of the layout, and produces report files that describe the accuracy in a comparison of the two results. When this option is specified, the .fscomptot and .fs_compcoup output comparison files always use the layout net names, regardless of the XREF command setting.
NORC	Retains only *|P and *|I information of the nets that are extracted or netlisted and the instance section with ideal connection for device terminals. Does not retain parasitic resistor or capacitor devices.

Description

The extraction of parasitic devices is performed only on that portion of the layout network defined by the NETS command, terminating each net at the boundary of a skip cell.

When you use the NORC option,

- The resistors and capacitors are not extracted if they are postprocessed with the NETLIST_POSTPROCESS_COMMAND command.
- The tool does not run the REDUCTION command if you use both the EXTRACTION: NORC and REDUCTION commands together. This is because the resistors and capacitors and their node information are not required in the netlist.
- The tool skips opens as they are not required with the EXTRACTION: NORC command.

Feedback

Examples

The following examples shows how the `EXTRACTION: NORC` command retains the *|P and *|I infromation of the nets in the generated netlist file.

```
starrc_shell> EXTRACTION: NORC

*|NET S0 0PF
*|P (S0 B 0 282.000 344.500)
*|I (I0|I18|M1@2:ABC I0|I18|M1@2 ABC B 0 287.500 306.000)
*|I (I0|I18|M1:ABC I0|I18|M1 ABC B 0 276.500 306.000)
*|I (I0|I18|M0:XYZ I0|I18|M0 XYZ B 0 287.500 284.000)
*|I (I0|I18|M0@2:XYZ I0|I18|M0@2 XYZ B 0 276.500 284.000)

*|NET CO 0PF
*|P (CO B 0 2.000 38.500)
*|I (I3|I19|M1:ABC I3|I19|M1 ABC B 0 26.500 51.000)
*|I (I3|I19|M1@2:ABC I3|I19|M1@2 ABC B 0 37.500 51.000)
*|I (I3|I19|M0:XYZ I3|I19|M0 XYZ B 0 26.500 29.000)
*|I (I3|I19|M0@2:XYZ I3|I19|M0@2 XYZ B 0 37.500 29.000)
```

See Also

- FSCOMPARE_OPTIONS
- FS_EXTRACT_NETS
- NETLIST_FORMAT
- NETLIST_POSTPROCESS_COMMAND
- REDUCTION

Feedback

FILL_SHORTS_LIMIT

Limits the number of fill shorts to include in summary reports and warning messages.

Syntax

```
FILL_SHORTS_LIMIT: max_count
```

Arguments

Argument	Description
max_count	Maximum number of fill shorts to report Default: 1000

Description

In cases where the StarRC tool identifies a large number of fill shorts, you might want to limit the amount of detail included in summary reports and the number of times that similar messages are issued.

The FILL_SHORTS_LIMIT command specifies the maximum number of unique fill shorts for which to report detailed information such as layer names and bounding boxes in the *shorts_all.sum* file. If the limit is exceeded, the StarRC tool writes a warning message in the file and does not report the additional shorts.

This command controls only fill shorts. For other types of shorts, use the SHORTS_LIMIT command.

See Also

- SHORTS_LIMIT
- Shorts Reports

Feedback

FSCOMPARE_COUPLING_RATIO

Specifies the minimum ratio of the coupling capacitance to the total capacitance required on a particular net to make it eligible for comparison in an `FSCOMPARE` extraction.

Syntax

```
FSCOMPARE_COUPLING_RATIO: value
```

Arguments

Argument	Description
value	Coupling capacitance ratio; a floating-point number between zero and one Default: 0.10

Description

The `FSCOMPARE_COUPLING_RATIO` command specifies the minimum ratio of the coupling capacitance to the total capacitance required on a particular net to make it eligible for comparison in an `FSCOMPARE` extraction. The results are filtered by the field solver results.

The specified value is applied to the capacitance values calculated by the field solver.

This command does not apply to field solver extractions specified by the `FS_EXTRACT_NETS` command.

Examples

```
FSCOMPARE_COUPLING_RATIO: 0.2
```

See Also

- EXTRACTION : FSCOMPARE
- FSCOMPARE_THRESHOLD
- FSCOMPARE_COUPLING_THRESHOLD

Feedback

FSCOMPARE_COUPLING_THRESHOLD

Specifies the minimum coupling capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction.

Syntax

```
FSCOMPARE_COUPLING_THRESHOLD: value
```

Arguments

Argument	Description
value	Coupling capacitance threshold Default: 1.0e-15

Description

The FSCOMPARE_COUPLING_THRESHOLD command specifies the minimum coupling capacitance required on a net to make it eligible for comparison in an FSCOMPARE extraction. The specified threshold is applied to the capacitance values calculated by the field solver.

The FSCOMPARE_COUPLING_THRESHOLD command applies only to coupling capacitances. Use the FSCOMPARE_THRESHOLD command to filter total capacitances.

This command does not apply to field solver extractions specified by the FS_EXTRACT_NETS command.

Examples

Set the FSCOMPARE_COUPLING_THRESHOLD command to 0 to specify that no nets are eliminated based on coupling capacitance values.

```
FSCOMPARE_COUPLING_THRESHOLD: 0
```

See Also

- EXTRACTION : FSCOMPARE
- FSCOMPARE_COUPLING_RATIO
- FSCOMPARE_THRESHOLD

Feedback

FSCOMPARE_FILE_PREFIX

Specifies the prefix to be attached to files generated by the `EXTRACTION: FSCOMPARE` command.

Syntax

`FSCOMPARE_FILE_PREFIX:` *prefix*

Arguments

Argument	Description
`prefix`	A prefix to be attached to the beginning of file names Default: block name specified in the StarRC command file

Description

This command specifies the prefix to be attached to files generated by the `EXTRACTION: FSCOMPARE` command.

This command does not apply to field solver extractions specified by the `FS_EXTRACT_NETS` command.

Examples

```
FSCOMPARE_FILE_PREFIX: myprefix
myprefix.fs-compcoup
```

See Also

- EXTRACTION

Feedback

FSCOMPARE_OPTIONS

Specifies field solver options such as convergence goal and multiprocessing.

Syntax

```
FSCOMPARE_OPTIONS: option_1 [option_2 …]
```

Arguments

Argument	Description
option_1 [*option_2* …]	Field solver options listed in Table 64

Description

Table 64 lists the options that you can specify as arguments in the FSCOMPARE_OPTIONS command. The options apply to all field solver extraction regardless of whether it is invoked with the FS_EXTRACT_NETS command or the EXTRACTION:FSCOMPARE command.

Table 64 Arguments of the FSCOMPARE_OPTIONS Command

Argument	Description
-f *input_files*	Specifies the input files. Specify the technology file before the design file.
-e *list_of_nets*	Specifies the list of nets to extract. Default: all nonground nets
-v	Prints the program version.
-np *number_processors*	For distributed processing, sets the number of processors to use. Default: 1
-nt *number_threads*	For multithreaded processing, sets the number of threads to use. This number of threads is used on each processor if more than one processor is enabled with the -np option. Default: 1 Maximum: 32
-time_out *wait_time*	For distributed processing, sets the maximum time that the supervisor process waits for the subordinate machines to start. Units: minutes Default: 1440

Feedback

Table 64 Arguments of the FSCOMPARE_OPTIONS Command (Continued)

Argument	Description
`-mach_term` *`retry_time`*	For distributed processing, sets the time to keep trying to contact a machine that has stopped responding. If the machine does not respond within the specified time, the job terminates. Units: minutes Default: 10
`-min_per_net` *`minutes_per_net`*	Sets the maximum extraction time per net Units: minutes Default = 20
`-l` *`file_name`*	Specifies the name of a file containing a list of client machines. For each client, specify a row with the following: *`machine_name arch`*. If `-l` is given, then LSF is not used for the clients.
`-perc_self` *`self_cap_conv_goal`*	Sets the self-capacitance convergence goal at one standard deviation as a percentage Units: percent Default: 0.5 for the `FSCOMPARE` flow 1.5 for the `FS_EXTRACT_NETS` flow
`-perc_coup` *`coup_cap_conv_goal`*	Sets the coupling capacitance convergence goal at one standard deviation as a percentage Default: not checking
`-abs_coup` *`abs_cap_conv_goal`*	Sets the absolute coupling capacitance convergence goal The dynamic value of *`abs_cap_conv_goal`* is determined by multiplying the total net capacitance by the percentage set by the `-coup_cap_thresh` option. Units: farads Default: dynamic
`-coup_cap_thresh` *`number`*	Sets the percentage of total capacitance at which to start checking coupling Units: percent Default: 1
`-perc_consistency` *`max_dev`*	Sets the maximum deviation between identical nets, expressed as a percentage of the total capacitance Units: percent Default: none
`-perc_of_nets_consistent` *`number`*	Sets the percentage of identical nets that deviate from each other by the level specified by the `-perc_consistency` option Units: percent Default: 97

Feedback

Table 64 Arguments of the FSCOMPARE_OPTIONS Command (Continued)

Argument	Description
`-perc_accuracy` *number*	Sets the accuracy of the extracted capacitance value, expressed as a percentage of the capacitance value. Note: if this parameter is specified, the `-perc_self` should not be specified. Units: percent Default: 1.5
`-perc_accuracy_confidence` *number*	Sets the confidence level for the estimated capacitance value, extracted at the accuracy level set by the `-perc_accuracy` option, expressed as a percentage Units: percent Default: 99.7
`-min_cap` *cap_value*	Sets the minimum capacitance value to report in the output file Units: farads Default: 1.0e-20
`-seed` *rand_num_seed*	Sets the random number seed Default: 12345
`-bb` *xl yl xh yh*	Sets the bounding box Units: grid units Default: 100 microns larger than design
`-neuman_x`	Uses Neumann boundary conditions on x-boundaries Default: false
`-neuman_y`	Uses Neumann boundary conditions on y-boundaries Default: false
`-periodic_x`	Uses periodic boundary conditions on x-boundaries Default: false
`-periodic_y`	Uses periodic boundary conditions on y-boundaries Default: false
`-match`	Enables pattern matching and improves runtime and accuracy for symmetric or identical net extraction in the field solver This option is supported in the field solver and QTF flows.

To obtain a list of the field solver options, enter `fieldsolver -help` on the command line.

The number of StarRC Ultra licenses checked out for a FSCOMPARE flow is less than or equal to NC*NP*NT/C, where NC is the number of cores specified by the `NUM_CORES`

Feedback

command, NP is the value specified by the `-np` option, NT is the value specified by the `-nt` option, and C is a constant. The value of C is 4 if the run uses features that require a StarRC Ultra license and 8 if the run does not use Ultra features.

Examples

To run two clients with a one percent self-capacitance goal, use the following command:

```
FSCOMPARE_OPTIONS: -np 2 -perc_self 1
```

To run four clients with a 10 percent coupling capacitance and one percent self-capacitance goal, use the following command:

```
FSCOMPARE_OPTIONS: -np 4 -perc_coup 10 -perc_self 1
```

See Also

- EXTRACTION : FSCOMPARE
- FS_EXTRACT_NETS
- FSCOMPARE_COUPLING_RATIO
- FSCOMPARE_COUPLING_THRESHOLD
- FSCOMPARE_THRESHOLD

Feedback

FSCOMPARE_THRESHOLD

Specifies the minimum total capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction.

Syntax

```
FSCOMPARE_THRESHOLD: value
```

Arguments

Argument	Description
value	Capacitance threshold Default: 3.0e-15

Description

FSCOMPARE_THRESHOLD specifies the minimum total capacitance required on a particular net to make it eligible for comparison in an FSCOMPARE extraction. The specified threshold is applied to the total capacitance values calculated by the field solver.

The FSCOMPARE_THRESHOLD command does not apply to coupling capacitances. Use the FSCOMPARE_COUPLING_THRESHOLD command to filter coupling capacitances.

Examples

Setting FSCOMPARE_THRESHOLD to 0 ensures that no nets are eliminated based on capacitance values.

```
FSCOMPARE_THRESHOLD: 0
```

See Also

- EXTRACTION : FSCOMPARE
- FS_EXTRACT_NETS
- FSCOMPARE_COUPLING_RATIO
- FSCOMPARE_COUPLING_THRESHOLD
- FSCOMPARE_OPTIONS

Feedback

FS_BOUNDARY_BOX

Overrides the default boundary box for the field solver.

Syntax

```
FS_BOUNDARY_BOX: x_min y_min z_min x_max y_max z_max
```

Arguments

Argument	Description
x_min y_min z_min x_max y_max z_max	Boundary box coordinates Units: microns

Description

The FS_BOUNDARY_BOX command overrides the default boundary box for the field solver.

The StarRC tool scales the x- and y- coordinates, but not the z-coordinates, of the boundary box when you specify either of the following:

- The MAGNIFICATION_FACTOR command in the StarRC command file
- The HALF_NODE_SCALE_FACTOR statement in the ITF file

Note:

Although you must specify *z_min* and *z_max* values, the tool ignores them.

Errors

If you do not specify the FS_BOUNDARY_BOX and FS_BOUNDARY_CONDITION commands together, the tool issues an error message.

Examples

In the following example, the design data is expressed as 10000 grids per user unit:

```
FS_BOUNDARY_BOX: 1.0 0.16 0.1 17.50 1.45 7.40
```

This command translates to the following setting, as reported in the “Invoking the Field Solver with Command” section of the StarRC summary file:

```
FSCOMPARE_OPTIONS: -bb 10000 1600 175000 14500
```

Alternatively, if you specify MAGNIFICATION_FACTOR: 2.0, then the previous FS_BOUNDARY_BOX command translates to the following setting:

```
FSCOMPARE_OPTIONS: -bb 20000 3200 350000 29000
```

Feedback

See Also

- FS_BOUNDARY_CONDITION
- FSCOMPARE_OPTIONS
- HALF_NODE_SCALE_FACTOR
- MAGNIFICATION_FACTOR

Feedback

FS_BOUNDARY_CONDITION

Overrides the default boundary conditions used in the field solver.

Syntax

```
FS_BOUNDARY_CONDITION:
  [-bc_xn N | P] [-bc_xp N | P] [-bc_yn N | P] [-bc_yp N | P]
```

Arguments

Argument	Description
`-bc_xn`	Boundary condition for the negative x-direction
`-bc_xp`	Boundary condition for the positive x-direction
`-bc_yn`	Boundary condition for the negative y-direction
`-bc_yp`	Boundary condition for the positive y-direction
`N`	Neumann boundary conditions
`P`	Periodic boundary conditions

Description

The `FS_BOUNDARY_CONDITION` command overrides the default boundary conditions used in the field solver.

You must specify

- The same boundary condition for the negative and positive x-directions
- The same boundary condition for the negative and positive y-directions

If you specify different boundary conditions for the positive and negative directions, the Neumann boundary condition overrides the periodic boundary condition.

Errors

If you do not specify the `FS_BOUNDARY_BOX` and `FS_BOUNDARY_CONDITION` commands together, the tool issues an error message.

Examples

The following examples specifies the use of Neumann boundary conditions on x-boundaries and periodic boundary conditions on y-boundaries:

```
FS_BOUNDARY_CONDITION: -bc_xn N -bc_xp N -bc_yn P -bc_yp P
```

This command translates to the following setting, as reported in the “Invoking the Field Solver with Commands” section of the StarRC summary file:

```
FSCOMPARE_OPTIONS: -neuman_x -periodic_y
```

See Also

- FS_BOUNDARY_BOX
- FSCOMPARE_OPTIONS

Feedback

FS_DP_STRING

Specifies the distributed processing method and job control parameters for field solver extraction runs.

Syntax

```
FS_DP_STRING:
      bsub lsf_arguments
    | qsub gridware_arguments
    | list list_of_machines
    | nc sub rtda_arguments
```

Arguments

Argument	Description
lsf_arguments	Arguments for an LSF system
gridware_arguments	Arguments for a Gridware system
list_of_machines	List of machines on a general network
rtda_arguments	Arguments for a Runtime Design Automation system

Description

The job submission command can be specified either in an environment variable or as a command in the StarRC command file. If set in both places, the StarRC command file takes precedence. The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Specify the number of processors and number of threads per processor to use for field solver distributed processing with the `FSCOMPARE_OPTIONS` command.

Distributed processing is available for the following computing environments:

- LSF system
- Gridware system
- General network with a list of machines
- Runtime Design Automation (RTDA) system

Examples

On an LSF system:

```
FS_DP_STRING: bsub -R "rusage[mem=5000]"
```

On a Gridware system:

```
FS_DP_STRING: qsub -P bnormal -l "mem_free=1G mem_avail=1G"
```

On a general network with a list of machines:

```
FS_DP_STRING: list alpha beta gamma
```

On an RTDA system:

```
FS_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

See Also

- FSCOMPARE_OPTIONS
- Distributed Processing for Field Solver Jobs

Feedback

FS_EXTRACT_NETS

Specifies nets to be extracted by the field solver.

Syntax

`FS_EXTRACT_NETS: net_names`

Arguments

Argument	Description
net_names	List of nets for field solver extraction. Wildcards can be used. Default: none (no nets)

Description

The FS_EXTRACT_NETS command specifies a list of nets for which field solver extraction should be used. The resulting netlist combines the nets extracted by standard StarRC pattern-matching extraction and the nets extracted by the field solver.

The StarRC tool creates comparison tables for both total and coupling capacitances with the FS_EXTRACT_NETS command. This enables you to generate an accurate parasitic netlist along with a validation report. This is available for all flows.

Use the FSCOMPARE_OPTIONS command to set up distributed processing for field solver extraction.

The following usage notes apply:

- The NET_TYPE:SCHEMATIC command is supported when used with the FS_EXTRACT_NETS command in the Calibre flow.
- Wildcards are supported: asterisk (*) for all values, question mark (?) for a single character, and exclamation mark (!) for negation.
- Net names that originate from a hierarchical netlist must be fully flattened with the hierarchical separator defined by the HIERARCHICAL_SEPARATOR command. Additionally, any reserved character from the input database must be included in this list to allow the use of special characters such as the BUS_BIT delimiter.
- Names must be case-sensitive in accordance with the CASE_SENSITIVE command.

The StarRC tool does not alter the net information in the input database except to flatten names as required. The case of names in the input database is always preserved in the output netlist. It is important to understand how the place and route tool handles names. If possible, extract names directly from the input database.

Feedback

Table 65 describes the behavior of the FS_EXTRACT_NETS command for different combinations of input that include wildcards.

Table 65 Behavior of Wildcards in the FS_EXTRACT_NETS Command

Argument	NETS command behavior
*	Selects all nets
* A B	Selects all nets
A	Selects only net A
XY*	Selects only nets with names beginning with XY
* !A	Selects all nets except net A
* !XY*	Selects all nets except nets with names beginning with XY

Examples

In the following example, the StarRC tool extracts all nets, but the field solver is used to extract the three nets specified by the FS_EXTRACT_NETS command:

```
NETS: *
FS_EXTRACT_NETS: net1 net2 net3
NET_TYPE: SCHEMATIC
```

See Also

- FSCOMPARE_OPTIONS
- NET_TYPE

Feedback

FS_IGNORE_GATE_CHANNEL_CAPACITANCE

For field solver analysis, restricts extraction of certain capacitances of a MOS device.

Syntax

```
FS_IGNORE_GATE_CHANNEL_CAPACITANCE: YES | NO
```

Arguments

Argument	Description
YES	Enables only outer fringe capacitance extraction
NO (default)	Performs standard extraction

Description

The FS_IGNORE_GATE_CHANNEL_CAPACITANCE command provides control over the extraction of gate to diffusion capacitances. The gate to diffusion capacitance usually includes both the capacitance between the gate conductor and the top surface of the diffusion region (outer fringe capacitance) and the capacitance between the gate conductor and the side of the diffusion region (inner fringe capacitance). Setting the command to YES specifies to extract only the outer fringe capacitance.

This command interacts with statements in the ITF file and other commands in the command file, as follows:

- If the ITF file contains a GATE_TO_DIFFUSION_CAP section, the FS_IGNORE_GATE_CHANNEL_CAPACITANCE command is overridden.

  The IGNORE_CAPACITANCE: ALL RETAIN_GATE_DIFFUSION_COUPLING command is also required.

- If the ITF file does not contain a GATE_TO_DIFFUSION_CAP section and the FS_IGNORE_GATE_CHANNEL_CAPACITANCE command is set to YES, only the outer fringe capacitance is extracted.

  The gate conductor must be identified as layer type GATE in the ITF file. The IGNORE_CAPACITANCE: ALL RETAIN_GATE_DIFFUSION_COUPLING command is also required.

- If the ITF file does not contain a GATE_TO_DIFFUSION_CAP section and the FS_IGNORE_GATE_CHANNEL_CAPACITANCE command is set to NO (or does not appear in the command file), the extracted gate to diffusion capacitance includes both the outer and inner fringe components.

Feedback

Examples

```
FS_IGNORE_GATE_CHANNEL_CAPACITANCE: YES
```

See Also

- EXTRACTION : FSCOMPARE
- FS_EXTRACT_NETS
- FSCOMPARE_OPTIONS
- GATE_TO_DIFFUSION_CAP
- IGNORE_CAPACITANCE

Feedback

FS_NON_MANHAT_RATIO

Specifies the ratio of stair-step size to the `WMIN` value for non-Manhattan geometries.

Syntax

```
FS_NON_MANHAT_RATIO: value
```

Arguments

Argument	Description
value	Ratio of stair-step size to `WMIN` value Default: 0.20

Description

The StarRC field solver analyzes non-Manhattan geometries by approximating polygon edges with small steps. The default is sufficient for most applications. If you reduce the value, accuracy improves at the cost of runtime.

See Also

- WMIN

Feedback

FS_QTF_FILE

Specifies a QTF file to model advanced device structures.

Syntax

```
FS_QTF_FILE: qtf_file
```

Arguments

Argument	Description
qtf_file	Name of the QTF file Default: none

Description

The FS_QTF_FILE command specifies the name of a QTF file, which is a process modeling file for advanced device structures and which is usually provided by a foundry.

QTF files are always processed by the StarRC field solver. Runtime for the QTF flow might be longer than for standard extraction flows.

See Also

- FS_QTF_OPTIONS
- map_qtf_layers
- qtf_layers
- The QTF Flow

Feedback

FS_QTF_OPTIONS

Specifies options for processing a QTF file with the StarRC field solver.

Syntax

```
FS_QTF_OPTIONS: option1
```

Arguments

Argument	Description
option1	Field solver options for QTF file analysis Default: None

Description

The `FS_QTF_OPTIONS` command specifies options that apply to field solver analysis of the QTF file. The primary usage of this command is to specify the reference direction for directional etches. For example, the following command sets the y-direction as the reference direction:

```
FS_QTF_OPTION: -qtfReference y
```

See Also

- FS_QTF_FILE
- map_qtf_layers
- qtf_layers
- The QTF Flow

Feedback

GDS_FILE

Specifies GDSII format files to represent part of the physical layout.

Syntax

```
GDS_FILE: file1 [file2] …
```

Arguments

Argument	Description
file1 [file2] …	Names of GDSII files containing physical layout information Default: none

Description

The GDS_FILE command specifies GDSII format files to represent part of the physical layout. You can specify gzip and compressed GDSII files for this command.

In the Fusion Compiler or IC Compiler II flow (NDM format designs) and IC Compiler flow (Milkyway format designs), the GDS_FILE command merges GDSII data into FRAM or CEL views for skip cells to provide a full physical layout representation. The StarRC tool uses only the pin shapes in the FRAM view for the cells that are defined both by the FRAM view and the GDSII file. The tool replaces obstructions with material defined within GDS cells of the same name. If no matching cell name is found within the GDSII file for a particular FRAM cell, the FRAM obstruction is used for that cell.

In the LEF/DEF flow, this command merges GDSII data into LEF MACRO definitions for skip cells to provide a full physical layout representation. The tool uses only the pin shapes from the LEF MACRO for the cells that are defined by both the LEF MACRO and the GDS file. Any material defined within the OBS section of the LEF MACRO is overwritten and replaced by material defined within the GDS cell of the same name. If no matching cell name is found, the OBS section of the corresponding LEF MACRO is used for that cell.

The GDS_FILE command can be specified multiple times. It must be used with the GDS_LAYER_MAP_FILE command, but cannot be used with the OASIS_FILE command.

See Also

- GDS_LAYER_MAP_FILE
- SKIP_CELLS
- The StarXtract -gdscheck Option

Feedback

GDS_LAYER_MAP_FILE

Specifies the mapping between the GDSII layer number and layer name in the design database.

Syntax

```
GDS_LAYER_MAP_FILE: file_name
```

Arguments

Argument	Description
file_name	The GDSII layer mapping file Default: none

Description

The GDS_LAYER_MAP_FILE command specifies the mapping between the GDSII layer number and layer name in the design database whenever the GDS_FILE or METAL_FILL_GDS_FILE command is used to import GDSII data into the design database.

Note:

You cannot use the GDS_LAYER_MAP_FILE command with the OASIS_LAYER_MAP_FILE command.

All translated GDSII layers must have an entry in the file specified by the GDS_LAYER_MAP_FILE command and must have a definition in the layout database. Use the GDS_LAYER_MAP_FILE command to import GDSII cell material into a Milkyway or LEF/DEF database or to import metal fill polygons into a Milkyway, LEF/DEF, or Calibre Connectivity Interface database. If you use the METAL_FILL_GDS_FILE and GDS_FILE commands in a single Milkyway or LEF/DEF run, use a unified layer mapping file for the GDSII files.

An error occurs if any layer is specified in the file does not have a corresponding layer in the layout database.

The GDS layer map file uses the following syntax:

```
database_layer gdsii_layer_number gdsii_datatype [MASK=mask_no]
                  [FLOATING | GROUNDED | IGNORE] [IP_FILL]
```

Argument	Description
database_layer	The database layer name.
gdsii_layer_number	The GDSII layer number.

Feedback

Argument	Description
gdsii_datatype	The GDSII data type. If a GDSII data type is not specified, then all data types on a given layer are read.
mask_no	The mask ID number for multimask patterning (an integer)
FLOATING	Specifies that the corresponding fill layer is to be treated as floating. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.
GROUNDED	Specifies that the corresponding fill layer is to be treated as grounded. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.
IGNORE	Specifies that the corresponding fill layer is to be ignored. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.
IP_FILL	Differentiates metal fills from other design information in the GDS layer map.

The layer-specific fill-handling keyword allows you to decide how individual metal fill layers are handled during parasitic extraction. This handling is considered only if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is set in the StarRC command file.

If the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command appears in the StarRC command file but a fill-handling mode is not specified for a layer in the GDS layer map file, the default setting is FLOATING for that layer. If another setting of the METAL_FILL_POLYGON_HANDLING command (other than AUTOMATIC) is specified, that setting governs the handling of all layers, and any layer-specific mode specifications inside the GDS layer map file are ignored.

In the example, handling mode GROUNDED is specified for layer DIFF. If the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is also specified in the StarRC command file, DIFF is treated as GROUNDED while all other layers are treated as FLOATING. If METAL_FILL_POLYGON_HANDLING:AUTOMATIC is not specified in the command file, the layer-specific mode specification for DIFF is ignored, and the global mode set by the METAL_FILL_POLYGON_HANDLING command takes precedence.

Examples

The following example shows how the DIFF layer is assigned to GDSII layer 2 and GDSII datatype 0. This example maps layers from a metal fill GDS file and specifies layer-specific fill handling for the DIFF layer.

Example 23 Layer-Specific Fill Handling

```
DIFF 2 0      GROUNDED
POLY 7 0
CONT 4 0
METAL1 10 0
METAL1 10 1
METAL1 76 0
VIA1 11 0
METAL2 12 0
```

In the following example, the `METAL_FILL_POLYGON_HANDLING: AUTOMATIC` command is set in the command file.

Example 24 Automatic Fill Handling

```
*layer treated as grounded
DIFF    2 0 GROUNDED
*layer treated as floating
POLY    7 0 FLOATING
*layer governed by default floating mode since mode is unspecified.
METAL1    4 0
```

See Also

- GDS_FILE
- LEF_FILE
- METAL_FILL_GDS_FILE

Feedback

GPD

Specifies the name of the directory in which to store binary parasitic data.

Syntax

`GPD: parasitics_dir`

Arguments

Argument	Description
parasitics_dir	The name of the parasitics database directory Default: *block*.gpd, where *block* is the top-level design block specified by the `BLOCK` command

Description

The GPD is a distributed and scalable parasitic database that is designed for efficient communication between the StarRC tool and the PrimeTime tool.

The GPD flow is enabled by default and this flow also supports transistor-level, field solver, clock net inductance, and power extraction flows.You can optionally use the `GPD` StarRC command to specify the name of the directory in which to store the parasitic data. The default is to use the name specified by the `BLOCK` command.

See Also

- GPD_DP_STRING
- NETLIST_LOCATION_TRANSFORMS
- The Parasitic Database or GPD

Feedback

GPD_CHECKS

GPD netlist checker that operates on the GPD to verify the output of a netlist in a gate-level flow.

Syntax

```
GPD_CHECKS: file_name
```

Description

For the following information, see GPD Netlist Checker

- To perform consistency check directly on the GPD without generating netlist files
- To validate parasitic information earlier in the StarRC flow

Feedback

GPD_DP_STRING

Specifies distributed processing conditions for use in a GPD configuration file. Valid only in a GPD configuration file.

Syntax

```
GPD_DP_STRING:
      bsub lsf_arguments
    | qsub gridware_arguments
    | list list_of_machines
    | list localhost num_processes
    | nc sub rtda_arguments
```

Arguments

Argument	Description
lsf_arguments	Arguments for an LSF system
gridware_arguments	Arguments for a Gridware system
list_of_machines	List of machines on a general network
num_processes	Number of processes on the local host
rtda_arguments	Arguments for a Runtime Design Automation system

Description

Use this command in a GPD configuration file to specify distributed processing conditions for netlist creation that are different from the distributed processing conditions used in the original extraction run. Distributed processing in the extraction run is controlled by the `STARRC_DP_STRING` command.

The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Distributed processing is available for the following computing environments:

- LSF system
- Gridware system
- General network with a list of machines
- Runtime Design Automation (RTDA) system

Feedback

Examples

On an LSF system:

```
GPD_DP_STRING: bsub -R "rusage[mem=5000]"
```

On a Gridware system:

```
GPD_DP_STRING: qsub -P bnormal -l "mem_free=1G mem_avail=1G"
```

On a general network with a list of machines:

```
GPD_DP_STRING: list alpha beta gamma
```

On an RTDA system:

```
GPD_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

See Also

- GPD
- The Parasitic Database or GPD
- STARRC_DP_STRING
- Distributed Processing

Feedback

GPD_IN_NDM

Attaches the GPD to an NDM format Fusion Compiler or IC Compiler II design.

Syntax

```
GPD_IN_NDM: YES | NO
```

Arguments

Argument	Description
YES	Attaches the GPD to an NDM format design
NO (default)	Does not attach the GPD to the design.

Description

Extracted parasitics are automatically saved in the binary parasitic database (GPD) for most flows. Downstream tools can read the GPD to obtain parasitics to annotate to a design.

For NDM format Fusion Compiler or IC Compiler II designs, you can save the GPD into the design database by setting the `GPD_IN_NDM` command to `YES`. This enables downstream tools to obtain the parasitics automatically when reading in the design.

For example, in PrimeTime version O-2018.06, using the `read_ndm` command reads both the design and the parasitics, removing the need to run the `read_parasitics -format gpd` command. To enable this capability, the StarRC command file must use the `GPD_IN_NDM: YES` command during the extraction run.

Setting this command to `NO` (the default) does not affect whether the GPD is created during StarRC extraction.

See Also

- GPD

Feedback

GRD_DP_MIN_CORES

Specifies the minimum number of cores to be available to proceed with a distributed processing run in the grdgenxo tool.

Syntax

```
GRD_DP_MIN_CORES: core_count
GRD_DP_MIN_CORES: core_pct%
```

Arguments

Argument	Description
core_count	Minimum number of cores, expressed as an integer Default: 1 Minimum: 1 Maximum: The value of the NUM_CORES command
core_pct%	Minimum percentage of cores . The percent symbol (%) is part of the syntax and must be included. Default: n/a (1 core)

Description

The GRD_DP_TIME_OUT and GRD_DP_MIN_CORES commands work together to define the computing resources necessary to proceed with a distributed processing run and the maximum amount of time to wait for those resources to become available.

Use the GRD_DP_MIN_CORES command to specify the minimum number of cores necessary to proceed with the grdgenxo run. You can set the value in either of the following ways:

- Specify an integer number of cores.
- Specify a percentage of the value of the NUM_CORES command:

  Minimum cores = NUM_CORES * *core_pct*/100

Note:

If you use the GRD_MIN_CORES command, you must also use the GRD_DP_TIME_OUT command or the run never begins. However, you can use the GRD_DP_TIME_OUT command alone, in which case the run proceeds when one core becomes available.

Feedback

See Also

- GRD_DP_STRING
- GRD_DP_TIME_OUT
- NUM_CORES
- Using Distributed Processing With the grdgenxo Tool

Feedback

GRD_DP_STRING

Enables automatic submission of distributed processing jobs in the grdgenxo tool.

Syntax

```
GRD_DP_STRING:
      bsub lsf_arguments
    | qsub gridware_arguments
    | list [login_protocol] host1[:n1] [host2[:n2] ... hostm[:nm]]
    | list localhost num_processes
    | nc run rtda_arguments
```

Arguments

Argument	Description
lsf_arguments	Arguments for an LSF system.
gridware_arguments	Arguments for a Gridware system.
login_protocol	Login protocol. Valid values: `rsh` (default) or `ssh`. Using the ssh protocol requires additional setup. For more information, see Distributed Processing.
host1, host2 ...	Name of host machine.
n1, n2 ...	Number of runs to submit on corresponding host.
num_processes	Number of processes on the local host.
rtda_arguments	Arguments for a Runtime Design Automation system (RTDA).

Description

Distributed processing allows you to start a single grdgenxo run and let the tool automatically submit multiple jobs. You can specify the job submission command by using the `GRD_DP_STRING` command in the DP configuration file.

The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Distributed processing is available for the following computing environments:

- A single host
- A general network with a list of machines
- LSF system

Feedback

- Gridware system
- Runtime Design Automation (RTDA) system

For more information about computing environments, see Distributed Processing.

Note:

The number of worker processes launched by the grdgenxo tool is equal to the setting of the NUM_CORES command. If your submission command specifies a larger number of cores, some cores are reserved but not used. For best results, the NUM_CORES command and the submission command should specify the same number of cores.

Examples

On an LSF system:

```
GRD_DP_STRING: bsub -R "rusage[mem=5000]"
```

On a Gridware system:

```
GRD_DP_STRING: qsub -P bnormal -l "mem_free=1G mem_avail=1G"
```

This example for a general network uses the ssh protocol and submits 4 runs on system alpha, 2 runs on system beta, and 1 run on system gamma:

```
GRD_DP_STRING: list ssh alpha:4 beta:2 gamma
```

On an RTDA system:

```
GRD_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

See Also

- ENABLE_IPV6
- NUM_CORES
- GRD_DP_MIN_CORES
- GRD_DP_TIME_OUT
- Using Distributed Processing With the grdgenxo Tool

Feedback

GRD_DP_TIME_OUT

Specifies the maximum time to wait for cores to become available in the grdgenxo tool.

Syntax

`GRD_DP_TIME_OUT: timeout`

Arguments

Argument	Description
max_count	Maximum time to wait for cores to become available Units: minutes Default: -1 (no timeout)

Description

The `GRD_DP_TIME_OUT` and `GRD_MIN_CORES` commands work together to define the computing resources necessary to proceed with a distributed processing run and the maximum amount of time to wait for those resources to become available.

If the `GRD_DP_TIME_OUT` command is set to zero, `-1` (the default), or any negative value, the wait time is infinite.

Note:

If you use the `GRD_MIN_CORES` command, you must also use the `GRD_DP_TIME_OUT` command or the run never begins. However, you can use the `GRD_DP_TIME_OUT` command alone, in which case the run proceeds when one core becomes available.

See Also

- GRD_DP_STRING
- GRD_DP_MIN_CORES
- Using Distributed Processing With the grdgenxo Tool

Feedback

GROUND_CROSS_COUPLING

Specifies whether to ground the small cross-coupling capacitances of a net with its neighboring orthogonal nets.

Syntax

```
GROUND_CROSS_COUPLING: YES | NO
```

Arguments

Argument	Description
YES (default for gate-level flows)	Grounds small cross-coupling capacitances
NO (default for transistor-level flows)	Retains small cross-coupling capacitances

Description

For gate-level flows, small cross-coupling capacitances between a net and its neighboring orthogonal nets are grounded by default. Set the GROUND_CROSS_COUPLING command to NO to retain and report these small capacitances.

For transistor-level flows, the capacitances are retained by default.

Feedback

HIERARCHICAL_COUPLING_ABS_THRESHOLD

Specifies an absolute threshold for reporting hierarchical coupling capacitances.

Syntax

```
HIERARCHICAL_COUPLING_ABS_THRESHOLD: threshold
```

Arguments

Argument	Description
threshold	Absolute threshold Units: farads (F) Default: 3e-15

Description

Hierarchical coupling capacitance occurs between extracted signal nets and skip cell instances. The hierarchical coupling capacitance includes the capacitance to all signal nets inside the instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

Hierarchical coupling capacitance is reported if the following conditions are all true:

- The `HIERARCHICAL_COUPLING_REPORT_FILE` command is present, which enables the feature and specifies a file name for the report.
- The absolute capacitance is larger than the value specified by the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` command.
- The relative capacitance, calculated as a percentage of the extracted signal net capacitance, is larger than the value specified by the `HIERARCHICAL_COUPLING_REL_THRESHOLD` command.
- The number of reported capacitances is smaller than the value specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Hierarchical capacitances are reported in decreasing order of their relative capacitance values.

See Also

- HIERARCHICAL_COUPLING_REPORT_NUMBER
- HIERARCHICAL_COUPLING_REPORT_FILE
- HIERARCHICAL_COUPLING_REL_THRESHOLD

Feedback

HIERARCHICAL_COUPLING_REL_THRESHOLD

Specifies the ratio of coupling to total capacitance.

Syntax

```
HIERARCHICAL_COUPLING_REL_THRESHOLD: threshold
```

Arguments

Argument	Description
threshold	Relative capacitance threshold; a floating point number between 0 and 1 Default: 0.03

Description

Hierarchical coupling capacitance occurs between extracted signal nets and skip cell instances. The hierarchical coupling capacitance includes the capacitance to all signal nets inside the instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

Hierarchical coupling capacitance is reported if the following conditions are all true:

- The `HIERARCHICAL_COUPLING_REPORT_FILE` command is present, which enables the feature and specifies a file name for the report.
- The absolute capacitance is larger than the value specified by the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` command.
- The relative capacitance, calculated as a percentage of the extracted signal net capacitance, is larger than the value specified by the `HIERARCHICAL_COUPLING_REL_THRESHOLD` command.
- The number of reported capacitances is smaller than the value specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Hierarchical capacitances are reported in decreasing order of their relative capacitance values.

See Also

- HIERARCHICAL_COUPLING_ABS_THRESHOLD
- HIERARCHICAL_COUPLING_REPORT_NUMBER
- HIERARCHICAL_COUPLING_REPORT_FILE

Feedback

HIERARCHICAL_COUPLING_REPORT_NUMBER

Specifies the number of nets to be reported in hierarchical coupling capacitance extraction.

Syntax

```
HIERARCHICAL_COUPLING_REPORT_NUMBER: no_of_nets
```

Arguments

Argument	Description
no_of_nets	Integer number of nets for which to report hierarchical coupling capacitors Default: 1000

Description

Hierarchical coupling capacitance occurs between extracted signal nets and skip cell instances. The hierarchical coupling capacitance includes the capacitance to all signal nets inside the instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

Hierarchical coupling capacitance is reported if the following conditions are all true:

- The `HIERARCHICAL_COUPLING_REPORT_FILE` command is present, which enables the feature and specifies a file name for the report.
- The absolute capacitance is larger than the value specified by the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` command.
- The relative capacitance, calculated as a percentage of the extracted signal net capacitance, is larger than the value specified by the `HIERARCHICAL_COUPLING_REL_THRESHOLD` command.
- The number of reported capacitances is smaller than the value specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Hierarchical capacitances are reported in decreasing order of their relative capacitance values.

See Also

- HIERARCHICAL_COUPLING_ABS_THRESHOLD
- HIERARCHICAL_COUPLING_REPORT_FILE
- HIERARCHICAL_COUPLING_REL_THRESHOLD

Feedback

HIERARCHICAL_COUPLING_REPORT_FILE

Enables hierarchical coupling capacitance analysis for gate-level extraction and specifies a file name for the report.

Syntax

```
HIERARCHICAL_COUPLING_REPORT_FILE: file_name
```

Arguments

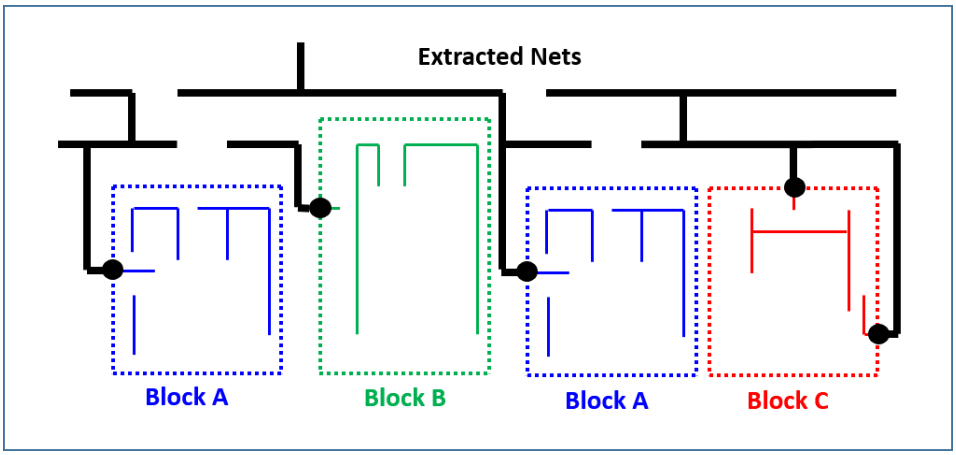
Argument	Description
file_name	Enables the hierarchical coupling capacitance feature and specifies a file name for the report Default: none

Description

The HIERARCHICAL_COUPLING_REPORT_FILE command enables the extraction of coupling capacitance between extracted signal nets and skipped cells and provides a name for the report.

Figure 198 illustrates a design that contains four skip cells. By default, features in skip cells are treated as ground during extraction. However, the coupling capacitance between extracted signal nets and skip cells can be significant and might affect timing analysis. Analyzing hierarchical coupling capacitance provides insight into this occurrence.

Figure 198 Relationship Between Extracted Nets and Skip Cell Nets



The hierarchical coupling capacitance includes the capacitance to all signal nets inside the skip cell instance but excludes the capacitance to other types of nets such as power, ground, blockage, and grounded fill nets.

The hierarchical coupling capacitance report presents coupling capacitances in decreasing order of the relative percentage of coupling capacitance to total capacitance for any extracted signal net. The report contains the number of entries specified by the `HIERARCHICAL_COUPLING_REPORT_NUMBER` command. Capacitances are filtered by the values of the `HIERARCHICAL_COUPLING_ABS_THRESHOLD` and `HIERARCHICAL_COUPLING_REL_THRESHOLD` commands.

In a simultaneous multicorner flow, the coupling report is generated only for the primary corner.

Examples

The following example shows the contents of a report file:

```
* 637 worst coupling capacitances listed in descending order:
* relative       coupling          extracted       instance
  weight         capacitance       net name        name
   23.2           1.386e-15        I237/B77        I20 (adder1)
   21.8           2.733e-15        T336/W22/I90    I31 (INV2)
   16.4           5.33e-14         PACKAGE_3       I20 (adder1)
   15.1           3.773e-15        GB_32772        I17 (mux2d4)
       ...
```

In this example, signal net I237/B77 has 23.2 percent of its total capacitance coupling to signal nets in skip cell instance I20. The value of the coupling capacitance is 1.386e-15 F.

See Also

- HIERARCHICAL_COUPLING_ABS_THRESHOLD
- HIERARCHICAL_COUPLING_REPORT_NUMBER
- HIERARCHICAL_COUPLING_REL_THRESHOLD

Feedback

HIERARCHICAL_SEPARATOR

Specifies the character used as the hierarchical delimiter during extraction and netlist creation.

Syntax

```
HIERARCHICAL_SEPARATOR: | | / | . | :
```

Arguments

Argument	Description
`|`	Pipe (|) character
`/` (default)	Slash (/) character
`.`	Period (.) character
`:`	Colon (:) character

Description

The `HIERARCHICAL_SEPARATOR` command specifies the character used as the hierarchical delimiter during extraction and netlist creation. If hierarchical nets are specified with the `NETS` command, this character must be used to derive flattened names for the selection.

Examples

This example sets the hierarchical separator to the period (.).

```
HIERARCHICAL_SEPARATOR: .
```

See Also

- BUS_BIT
- NETS

Feedback

HIGH_CLOCK_NET_FREQUENCY

Specifies the upper analysis frequency for clock net inductance extraction.

Syntax

```
HIGH_CLOCK_NET_FREQUENCY: freq
```

Arguments

Argument	Description
freq	Upper clock frequency, in GHz Default: Value specified by the CLOCK_NET_FREQUENCY command Range: Value of the CLOCK_NET_FREQUENCY command to 50 GHz

Description

The HIGH_CLOCK_NET_FREQUENCY command specifies the upper analysis frequency for clock net inductance analysis. To use this command, the CLOCK_NET_INDUCTANCE command must be set to YES.

The frequency specified by this command is taken into account as follows:

- If the CLOCK_NET_ADVANCED_MODEL command is set to NO (the default), the StarRC tool calculates one resistance value at zero frequency and one inductance value for each net segment at the frequency specified by the CLOCK_NET_FREQUENCY command.
- If the CLOCK_NET_ADVANCED_MODEL command is set to YES and the HIGH_CLOCK_NET_FREQUENCY command is not set, the tool calculates one resistance value and one inductance value for each net segment at the frequency specified by the CLOCK_NET_FREQUENCY command.
- If the CLOCK_NET_ADVANCED_MODEL command is set to YES and the HIGH_CLOCK_NET_FREQUENCY command is set to a value larger than the CLOCK_NET_FREQUENCY command value, the tool calculates frequency-dependent resistance and inductance values for each net segment. The result is a netlist that can be used for frequency analysis in downstream simulation tools. The model is valid between the two specified frequencies.

See Also

- CLOCK_NET_ADVANCED_MODEL
- CLOCK_NET_FREQUENCY
- Clock Net Inductance Extraction

Feedback

HIGH_R_LAYERS

Specifies layers that always generate physical resistors.

Syntax

```
HIGH_R_LAYERS: db_layer1 db_layer2 ...
```

Arguments

Argument	Description
db_layer1 ...	Layers that always generate physical resistors Default: none

Description

Use the `HIGH_R_LAYERS` command to specify layers to be treated as regular conductor layers even if their RPSQ values in the mapping file are very small. Resistors on these layers are always reported as physical resistors and their properties such as bounding box coordinates are saved in the GPD or reported in the output netlist.

If the specified layers are not present in the design, the StarRC tool ignores the command.

Feedback

HI_R_THERM_DEVICES

Specifies high-resistance devices to consider for thermal effects to generate parasitics for electromigration analysis.

Syntax

```
HI_R_THERM_DEVICES: hr_device1 hr_device2 ...
```

Arguments

Argument	Description
hr_device1 ...	Devices to consider for thermal effects due to high resistance Default: none

Description

Thermal effects are important in electromigration analysis. The StarRC tool provides two commands to specify high-resistance devices that might cause thermal issues. This improves the accuracy of the output netlist for use in downstream electromigration analysis.

Use the following commands in the StarRC command file to specify high-resistance devices:

- Use the `HI_R_THERM_DEVICES` command to list all devices to consider for thermal effects.
- Use the `HI_R_THERM_EXTENSION` command to specify the distance beyond the high-resistance device body that defines the high-resistance area bounding box. The default is 0.5 um.

The high-resistance regions do not affect extraction results. For these regions, the StarRC tool creates additional resistors and nodes for polygons that meet the following requirements:

- The polygons are on layers equal to or higher than the high-resistance device layer
- The polygons overlap the high-resistance device bounding box (abutting or touching does not count)

If the specified devices are not present in the design, the tool ignores the command.

See Also

- HI_R_THERM_EXTENSION

Feedback

HI_R_THERM_EXTENSION

Specifies a modification to high-resistance devices for electromigration analysis.

Syntax

```
HI_R_THERM_EXTENSION: hr_extension
```

Arguments

Argument	Description
hr_extension	Distance to define the bounding box for high-resistance devices Units: microns Default: 0.5

Description

Thermal effects are important in electromigration analysis. The StarRC tool provides two commands to specify high-resistance devices that might cause thermal issues. This improves the accuracy of the output netlist for use in downstream electromigration analysis.

Use the following commands in the StarRC command file to specify high-resistance devices:

- Use the `HI_R_THERM_DEVICES` command to list all devices to consider for thermal effects.
- Use the `HI_R_THERM_EXTENSION` command to specify the distance beyond the high-resistance device body that defines the high-resistance area bounding box. The default is 0.5 um.

The high-resistance regions do not affect extraction results. For these regions, the StarRC tool creates additional resistors and nodes for polygons that meet the following requirements:

- The polygons are on layers equal to or higher than the high-resistance device layer
- The polygons overlap the high-resistance device bounding box (abutting or touching does not count)

If the specified devices are not present in the design, the StarRC tool ignores the command.

See Also

- HI_R_THERM_DEVICES

Feedback

HN_NETLIST_MODEL_NAME

Writes a simulation model name instead of the StarRC model name in the parasitic netlist.

Syntax

```
HN_NETLIST_MODEL_NAME: ref_model_name SPICE_model_name
```

Arguments

Argument	Description
ref_model_name	The model name in the schematic or layout; no wildcards allowed Default: none
SPICE_model_name	The internal database is updated with this SPICE model name; no wildcards allowed Default: none

Description

This command updates the internal database with the model name provided in the argument list. The command also controls the model names of devices in an ideal netlist with the NETLIST_IDEAL_SPICE_FILE command. The MODEL_TYPE command determines whether the StarRC tool looks for a reference model in the layout or schematic netlist.

If the specified model is not present, the tool issues a warning message. If the same model name is specified more than one time, the last command overrides earlier commands.

You can map multiple reference models to a single simulation model.

If you specify the XREF_USE_LAYOUT_DEVICE_NAME and HN_NETLIST_MODEL_NAME command in the same command file, the HN_NETLIST_MODEL_NAME setting overrides the XREF_USE_LAYOUT_DEVICE_NAME setting.

Examples

If you have multiple device models, specify the command as follows:

```
HN_NETLIST_MODEL_NAME: p_dev pmos
HN_NETLIST_MODEL_NAME: pdev2 pmos-2
```

See Also

- MODEL_TYPE
- NETLIST_IDEAL_SPICE_FILE

Feedback

HN_NETLIST_SPICE_TYPE

Specifies StarRC netlist standard devices as SPICE .SUBCKT calls beginning with X.

Syntax

```
HN_NETLIST_SPICE_TYPE: model_name X
```

Arguments

Argument	Description
model_name	For a Hercules flow, a layout model name specified in a device extraction command, or a schematic model name specified in an EQUATE statement. For a Calibre flow, a model name or netlist model name specified in a DEVICE command. Default: none

Description

Specifies StarRC netlist standard devices as SPICE .SUBCKT calls beginning with X. When you specify this command for a standard, nonuser-defined ideal device, the SPICE device card is replaced with an X card in the netlist.

For cases in which multiple device extraction commands match a single device model name specified with the HN_NETLIST_SPICE_TYPE command, X device names are included in the netlist for devices from all matching commands.

In a Hercules flow, the SPICE device name can be set directly in the Hercules runset using the DEVICE_PREFIX option. This setting is propagated into the output netlist even if you do not use the HN_NETLIST_SPICE_TYPE command. The Hercules DEVICE_PREFIX option is defined using a string argument. The netlist includes only the first character of that string argument. See the Hercules documentation for more information.

In a Calibre flow, the device model name is read from block.devtab for model name and model card information:

```
DEVICE {element_name [(model_name)]} device_layer…
[NETLIST MODEL netlist_model_name] [NETLIST ELEMENT
netlist_element_name]
[[property_specification]]
```

The StarRC tool treats an element or model name as a layout name and a netlist element name as a schematic name.

Examples

```
HN_NETLIST_SPICE_TYPE: p X
```

Feedback

ICV_ANNOTATION_FILE

Specifies the IC Validator annotation file. Valid only for transistor-level extraction using an IC Validator flow.

Syntax

```
ICV_ANNOTATION_FILE: file_name
```

Arguments

Argument	Description
file_name	Name of the gzip-format IC Validator annotation file Default: adp.gz

Description

The `ICV_ANNOTATION_FILE` command specifies the IC Validator annotation file. This annotation file is produced by IC Validator to extract advanced device properties efficiently such as the well-proximity effect (WPE) and the length of diffusion (LOD). This command also triggers the IC Validator Dual Hierarchy Extraction (DHE) flow. You do not need the `ICV_ANNOTATION_FILE` command in the StarRC command file if the `annotation_file` statement exists in your IC Validator runset report file.

The IC Validator annotation file must be compressed into the gzip format.

In this annotation file, a line starting with the asterisk (*) character is considered a comment and ignored. The annotation file contains device property data in the SPICE format.

Examples

```
ICV_ANNOTATION_FILE: ./my_icv_adp.gz
```

Example 25 Syntax of the IC Validator Annotation File

```
- Define File
property_annotation_file (
file = "string" //optional
);
- Initiate DHE Flow
init_device_matrix(dual_hierarchy_extraction=true)
- Write Annotation file
write_annotation_file (
device_db = device_database,
output_file = property_annotation_file_handle,
precision = integer //optional
);
```

Feedback

Example 26 Example of an IC Validator Annotation File

```
.SUBCKT mc_co_stld_4x8
M0 sa =0.11u sa1=1.1e-07 sa2=1.1e-07 sa3=1.1e-07 sb=0.29u sb1=2.9e-07
sb2=2.9e-07 sb3=2.9e-07
M1 sa=0.11u sa1=1.1e-07 sa2=1.1e-07 sa3=1.1e-07 sb=0.11u sb1=1.1e-07
sb2=1.1e-07 sb3=1.1e-07
.ENDS

.SUBCKT blkcontrolc
M3  sa=0.29u sa1=2.9e-07 sa2=2.9e-07 sa3=2.9e-07 sb=1.19u sb1=1.19e-06
sb2=1.19e-06 sb3=1.19e-06
M4 sa=1.01u sa1=1.01e-06 sa2=1.01e-06 sa3=1.01e-06 sb=0.47u sb1=4.7e-07
sb2=4.7e-07 sb3=4.7e-07 sca=1.19602 scb=1.91168e-06 scc=2.37977e-12
spa=1.4e-07 spa1=1.4e-07 spa2=1.4e-07 spba=1e-06
X1/M2 sa=1.01u sa1=1.01e-06 sa2=1.01e-06 sa3=1.01e-06 sb=0.47u
sb1=4.7e-07 sb2=4.7e-07 sb3=4.7e-07 sca=1.19602 scb=1.91168e-06
scc=2.37977e-12 spa=1.4e-07 spa1=1.4e-07 spa2=1.4e-07 spba=1e-06
.ENDS
```

See Also

- ICV_RUNSET_REPORT_FILE

Feedback

ICV_LVS_DEVICE_TYPE_CAP

Identifies user-defined capacitance devices for transistor-level extraction using the IC Validator flow.

Syntax

```
ICV_LVS_DEVICE_TYPE_CAP: list_of_C_devices
```

Arguments

Argument	Description
list_of_C_devices	List of user-defined capacitance devices

Description

This command identifies user-defined capacitors as CAP devices.

The *list_of_C_devices* argument follows the case-sensitivity set by the CASE_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts in certain situations.

Examples

```
ICV_LVS_DEVICE_TYPE_CAP: cap_ss
```

See Also

- ICV_LVS_DEVICE_TYPE_MOS
- ICV_LVS_DEVICE_TYPE_RES
- ICV_OPTIONAL_DEVICE_PIN_FILE

Feedback

ICV_LVS_DEVICE_TYPE_MOS

Identifies user-defined MOS devices for transistor-level extraction using the IC Validator flow.

Syntax

```
ICV_LVS_DEVICE_TYPE_MOS: list_of_M_devices
```

Arguments

Argument	Description
list_of_M_devices	List of user-defined MOS devices

Description

This command identifies user-defined MOS devices.

The *list_of_M_devices* argument follows the case-sensitivity set by the CASE_SENSITIVE command and must use a layout name. Using schematic names might cause conflicts in certain situations.

Examples

```
ICV_LVS_DEVICE_TYPE_MOS: nmos_ss
```

See Also

- ICV_LVS_DEVICE_TYPE_CAP
- ICV_LVS_DEVICE_TYPE_RES
- ICV_OPTIONAL_DEVICE_PIN_FILE

Feedback

ICV_LVS_DEVICE_TYPE_RES

Identifies user-defined resistors as RES devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command. Valid for transistor-level extraction using the IC Validator flow.

Syntax

```
ICV_LVS_DEVICE_TYPE_RES: list_of_R_devices
```

Arguments

Argument	Description
list_of_R_devices	List of user-defined RES devices

Description

The `ICV_LVS_DEVICE_TYPE_RES` statement identifies user-defined resistors as RES devices and marks the device terminal layers for recognition by the `WIDE_DEVICE_TERM_RESISTANCE` command.

Examples

In the following example, the rp_sio and pwr_rm1 devices defined in the rule file are identified as resistors:

```
ICV_LVS_DEVICE_TYPE_RES:  rp_sio  pwr_rm1
```

See Also

- ICV_LVS_DEVICE_TYPE_CAP
- ICV_LVS_DEVICE_TYPE_MOS
- ICV_OPTIONAL_DEVICE_PIN_FILE
- WIDE_DEVICE_TERM_RESISTANCE

Feedback

ICV_OPTIONAL_DEVICE_PIN_FILE

Assigns nonstandard device terminals for transistor-level extraction in the IC Validator flow.

Syntax

```
ICV_OPTIONAL_DEVICE_PIN_FILE: file_name
```

Arguments

Argument	Description
file_name	Name of the device pin file

Description

By default, the StarRC tool assigns nonstandard device terminals by position. However, if you specify the `ICV_OPTIONAL_DEVICE_PIN_FILE` command, the tool assigns the device terminal by the name in the specified file.

The syntax of each line in the file is as follows:

```
device_type model_name[seed_layer] pin_layer_1 (pin name 1) \
     pin_layer_2 (pin name 2) pin_layer_3 (pin name 3) ...
```

Valid values for the `device_type` parameter are M (MOS device), C (capacitor), and R (resistor).

The seed layer is optional, but it must be used when multiple devices have the same model name.

Examples

In the following example, devpin_file contains the list of device terminal names:

```
ICV_OPTIONAL_DEVICE_PIN_FILE: devpin_file
```

The following lines show an example of a device pin file:

```
M MOS_DEV NDIFSI_D (D) POLY  (G) NDIFSI_S (S) NWELL (B)
C CAP_DEV CAP_TOP (PLUS)CAP_BOT (MINUS)
```

See Also

- ICV_LVS_DEVICE_TYPE_CAP
- ICV_LVS_DEVICE_TYPE_MOS
- ICV_LVS_DEVICE_TYPE_RES

Feedback

ICV_RUNSET_REPORT_FILE

Specifies the IC Validator runset report file and activates the transistor-level IC Validator flow.

Syntax

```
ICV_RUNSET_REPORT_FILE: file_name
```

Arguments

Argument	Description
file_name	IC Validator runset report file name Default: pex_runset_report

Description

The IC Validator runset report file contains information that the IC Validator tool provides to the StarRC tool for RC extraction, such as connection information, the location of LVS COMPARE output, device pin and properties information, and location of the extract view.

When the `ICV_RUNSET_REPORT_FILE` command is used, the `MILKYWAY_EXTRACT_VIEW` command is set to `YES`. In addition, the `MILKYWAY_DATABASE`, `MAPPING_FILE`, `COMPARE_DIRECTORY`, and `OA_LAYER_MAPPING_FILE` commands become optional. If these optional commands are specified in the StarRC command file, then the values in the StarRC command file override the values in the IC Validator runset report file.

The StarRC tool automatically reads different formats of IC Validator data. The IC Validator tool provides information about the data format in the runset report file.

Examples

```
ICV_RUNSET_REPORT_FILE: my_icv_rrf
```

See Also

- COMPARE_DIRECTORY
- MAPPING_FILE
- MILKYWAY_DATABASE
- MILKYWAY_EXTRACT_VIEW
- OA_LAYER_MAPPING_FILE

Feedback

IGNORE_BUMP_CELL_PINS

Specifies whether to prevent bump cell pins from appearing in the output netlist.

Syntax

```
IGNORE_BUMP_CELL_PINS: YES | NO
```

Arguments

Argument	Description
`NO` (default)	Reports bump cell pins in the output netlist
`YES`	Prevents bump cell pin reporting in the output netlist

Description

Bumps are physical features whose purpose is to connect chips to packages or to other chips. A bump cell typically contains bump geometries only on the topmost metal layer of the chip. In addition, a bump cell might also contain a via and a pin to the metal layer below.

Bump cell pins in a SPEF netlist sometimes cause annotation problems in downstream tools because the bump cells are not included in the Verilog design. To prevent bump cell pins from appearing in a SPEF netlist, set the `IGNORE_BUMP_CELL_PINS` command to `YES`.

The StarRC tool recognizes bump cells as follows:

- LEF/DEF designs

  The LEF macro file contains the CLASS COVER BUMP statement.

- NDM format Fusion Compiler or IC Compiler II designs

  The cell type is NDM_DESIGN_TYPE_FLIP_CHIP_PAD.

Feedback

IGNORE_CAPACITANCE

Prevents certain types of device-level capacitances from being extracted. Valid only for transistor-level extraction.

Syntax

```
IGNORE_CAPACITANCE: ALL [RETAIN_GATE_DIFFUSION_COUPLING] | DIFF | NONE
```

Arguments

Argument	Description
`ALL` (default)	Ignores capacitance between diffusion regions and the substrate, as well as between the transistor gates and diffusion regions or substrate. Both overlap and sidewall effects are ignored.
`RETAIN_GATE_DIFFUSION_COUPLING`	Retains gate-to-diffusion coupling capacitance
`DIFF`	Ignores only the junction capacitance. The gate capacitance is included.
`NONE`	Includes all parasitic capacitance

Description

The `IGNORE_CAPACITANCE` command prevents certain types of device-level capacitances from being extracted. This prevents double counting when the primitive device models already account for those capacitances.

The `IGNORE_CAPACITANCE` command applies to capacitive effects internal to individual devices. The `IGNORE_CAPACITANCE: ALL` and `IGNORE_CAPACITANCE: DIFF` commands do not exclude capacitive effects between neighboring devices because these effects are not part of the device model.

Feedback

Figure 199 labels the capacitances within a transistor and between transistors. Table 66 describes which capacitances are extracted for each command setting.

Figure 199 Capacitance Components of the IGNORE_CAPACITANCE Command

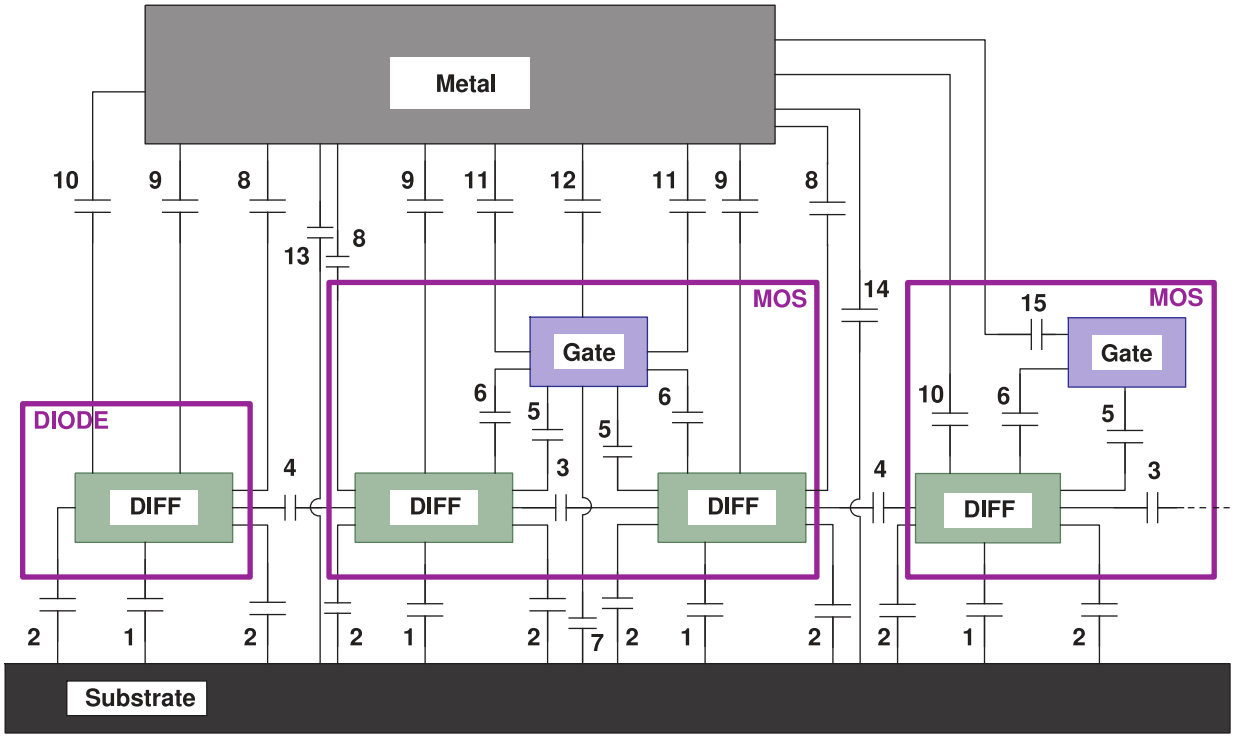


Table 66 Capacitances Extracted for Each Argument

Argument	Extracted capacitances	Ignored capacitances
ALL	4; 8 to 15	1 to 3; 5 to 7
DIFF	4 to 15	1 to 3
NONE	1 to 15	none

The StarRC field solver does not extract capacitances 1 to 3 and 5 to 7 for FinFET devices during FinFET process characterization. Therefore, the `DIFF` and `NONE` arguments are the same for these devices.

FinFET models that do not use the `MULTIGATE_MODELS: YES` statement already account for capacitances 1 to 3. For these models, using `DIFF` instead of `NONE` prevents double counting of those capacitances.

Feedback

The following example contains two MOS devices:

```
NMOS N ngate nsd nsd SUBSTRATE { } TEMP=ndev_out

PMOS P pgate psd psd NWELL { } TEMP=pdev_out
```

Table 67 lists the command interactions for the previous example.

Table 67 Command Interactionsfor IGNORE_CAPACITANCE: ALL

Ignored interactions for IGNORE_CAPACITANCE: ALL	Retained interactions for IGNORE_CAPACITANCE: ALL
ngate-SUBSTRATEnsd-nsdngate-nsdnsd-SUBSTRATEpgate-NWELLpsd-psdpgate-psdpsd-NWELL	ngate-pgatensd-psdngate-psdpgate-nsdngate-NWELLnsd-NWELL

This means that interdevice capacitive effects that are not accounted for in the device model are included in the parasitic netlist. For `IGNORE_CAPACITANCE` to be most effective and accurate, it is very important that device layers in the runset be separated from other connected layers that conflict with them (such as in the CONNECT sequence of the runset).

For example, if nsd and psd are derived from input layers DIFF, PPLUS and NPLUS, and DIFF is also a final CONNECT layer, the following Boolean sequence is required:

- BOOLEAN DIFF not nsd { } TEMP=DIFF
- BOOLEAN DIFF not psd { } TEMP=DIFF

This is critical because the DIFF layer is not used in a device and therefore it is not identifiable as a diffusion layer. Therefore, its parasitic contribution to both N and P devices cannot be ignored. In other words, DIFF is not an N or P device layer, so it must be part of the unmodeled environment.

When the calculation is done for netlist reduction to reduce the nodes with error control, small coupling capacitors are moved around their individual nodes to attach to one of the neighboring nodes. In such a situation, some device terminal nodes get coupling capacitances even if the coupling capacitance is not associated with them irrespective of the `IGNORE_CAPACITANCE` setting.

Retaining Gate-to-Diffusion Coupling Capacitance

To retain the gate-to-diffusion (Cf) coupling component when `IGNORE_CAPACITANCE:ALL` is specified, use the `RETAIN_GATE_DIFFUSION_COUPLING` keyword.

```
IGNORE_CAPACITANCE: ALL RETAIN_GATE_DIFFUSION_COUPLING
```

Feedback

When this keyword is specified, the StarRC tool has two methods for extracting the gate-to-diffusion component:

- Based on precharacterized models, similar to other extracted capacitances
- Based on a 2-D capacitance table lookup dependent on layout parameters

FinFET Capacitance

Table 68 lists the FinFET capacitance components retained when you use the `IGNORE_CAPACITANCE` command with `MULTIGATE` devices.

The difference between the `DIFF` and `NONE` keywords is that diffusion-to-substrate capacitance (Cd0) is extracted for finFETs when the `MULTIGATE_MODELS: YES` command is used in the StarRC command file.

Table 68 FinFET Capacitance Components Retained

IGNORE_CAPACITANCE Value	FinFET Capacitance Components Retained
`ALL` (default)	Cgc + Cfpe + Cfpc
`ALL RETAIN_GATE_DIFFUSION_COUPLING`	Cgc + Cfpe + Cfpc + Cgd
`DIFF`	Cgc + Cfpe + Cfpc + Cgd + Cg0
`NONE`	Cgc + Cfpe + Cfpc + Cgd + Cg0 + Cd0

Feedback

IGNORE_FIELDPOLY_DIFFUSION_COUPLING

Specifies whether to ignore or retain field poly to diffusion capacitance. Valid only for transistor-level flows.

Syntax

```
IGNORE_FIELDPOLY_DIFFUSION_COUPLING: YES | NO
```

Arguments

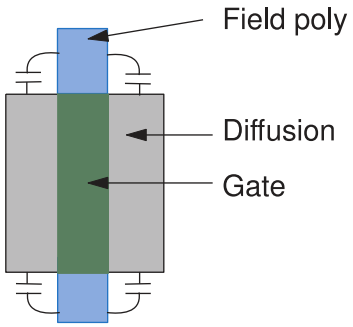
Argument	Description
NO (default)	Retains field poly to diffusion capacitance
YES	Excludes field poly coupling capacitance

Description

The IGNORE_FIELDPOLY_DIFFUSION_COUPLING command affects the field poly to diffusion capacitance shown in Figure 200. Although this orthogonal capacitance is typically small, at advanced technology nodes it might be important due to the large number of field poly connections in a design.

This command provides the flexibility to ignore field poly to diffusion capacitance during extraction when this effect is already included in the SPICE model.

Figure 200 Field Poly to Diffusion Capacitance Retained by Default



See Also

- IGNORE_CAPACITANCE

Feedback

IGNORE_GATE_CHANNEL_CAPACITANCE

Specifies the method of calculating gate-to-diffusion capacitance for FinFET devices.

Syntax

```
IGNORE_GATE_CHANNEL_CAPACITANCE: YES | NO
```

Arguments

Argument	Description
`NO` (default)	Disables use of Cfi table to calculate Cfo
`YES`	Enables calculation of Cfo by subtracting Cfi table values from extracted Cf values

Description

The total gate-to-diffusion capacitance (Cf) is the sum of the gate-to-diffusion capacitance inside the channel (Cfi) and the gate-to-diffusion capacitance outside the channel (Cfo).

You can provide a table to represent gate-to-diffusion capacitance inside the channel (Cfi) by using the ITF `GATE_TO_DIFFUSION_CHANNEL_CAP` statement.

When the `IGNORE_GATE_CHANNEL_CAPACITANCE: NO` command is used, the total gate-to-diffusion capacitance (Cf) is extracted by the field solver and no Cfi capacitance subtraction occurs, regardless of whether a Cfi table exists.

Alternatively, you can calculate Cfo by first having the field solver extract the total capacitance Cf, then subtracting the Cfi value (interpolated from the Cfi table) from Cf. To enable this calculation method, use the `IGNORE_GATE_CHANNEL_CAPACITANCE: YES` command. If a Cfi table does not exist, the command has no effect.

See Also

- GATE_TO_DIFFUSION_CHANNEL_CAP
- IGNORE_CAPACITANCE

Feedback

IGNORE_SHARED_MOS_TERMINAL_CAP

Ignores capacitance on shared terminals.

Syntax

```
IGNORE_SHARED_MOS_TERMINAL_CAP: YES | NO
```

Arguments

Argument	Description
`YES`	Ignores capacitance on a terminal layer shared by a MOS device terminal and a capacitor device terminal or on a terminal layer shared by a MOS device terminal and the resistance body of a resistor device
`NO` (default)	Extracts capacitance on a terminal layer shared by a MOS device terminal and a capacitor device terminal or on a terminal layer shared by a MOS device terminal and the resistance body of a resistor device

Description

By default, the StarRC tool extracts the capacitance between the following layers if a gate, diffusion, or bulk terminal is used as a capacitor terminal:

- The substrate and the gate, diffusion, and bulk layers
- The bulk layers and the gate and diffusion layers

To ignore this capacitance, specify `IGNORE_SHARED_MOS_TERMINAL_CAP: YES`.

Examples

In the following example, the tool ignores the capacitance on shared terminals.

```
IGNORE_SHARED_MOS_TERMINAL_CAP: YES
```

See Also

- IGNORE_CAPACITANCE

Feedback

INCLUDE_FILE

Specifies a file containing StarRC commands.

Syntax

```
INCLUDE_FILE: file_name
```

Arguments

Argument	Description
file_name	The name of a file containing StarRC commands Default: none

Description

A StarRC command file is a list of commands that specify conditions for the extraction run. You can specify multiple command files with the `StarXtract` command. In addition, you can use the `INCLUDE_FILE` command inside a command file to specify a separate file that contains StarRC commands.

For example, you might want to include general setup commands in one command file and design-specific commands in a separate file.

Commands in separate command files are treated as if they were written in a single command file, in the order provided. The first command file is read first, followed by the second command file, and so on. If any command file contains an `INCLUDE_FILE` command, the specified file is read immediately.

If StarRC commands are duplicated, later instances of a command usually overwrite earlier instances. However, some commands that contain lists of objects are cumulative. For information about how the StarRC tool treats multiple instances of specific commands, see the reference pages for individual commands.

See Also

- The StarRC Command File

INCLUDE_PG_IN_VERILOG

Specifies whether to include power and ground nets in the Parasitic Explorer analysis.

Syntax

```
INCLUDE_PG_IN_VERILOG: YES | NO
```

Feedback

Arguments

Argument	Description
YES (default for transistor-level flow)	Includes power and ground nets in parasitic analysis.
NO (default for gate-level flow)	Does not include power and ground nets in parasitic analysis.

Description

To ensure that the Parasitic Explorer tool has the required information, you must set the PARASITIC_EXPLORER_ENABLE_ANALYSIS command to YES to include power and ground nets in the Parasitic Explorer analysis.

For more information, see the *Parasitic Explorer User Guide*.

See Also

- PARASITIC_EXPLORER_ENABLE_ANALYSIS

Feedback

INDESIGN_OPEN_NETS

Specifies nets that should not be included in reduction operations.

Syntax

```
INDESIGN_OPEN_NETS: net_names
```

Arguments

Argument	Description
net_names	List of nets to exclude from reduction. Wildcards can be used. Default: none

Description

The INDESIGN_OPEN_NETS command specifies a list of nets that should not be included in reduction operations. This command provides compatibility with the Fusion Compiler or IC Compiler II In-Design extraction methodology for handling open nets.

See Also

- REDUCTION

Feedback

INDESIGN_VIRTUAL_SHIELDING

Models virtual shielding for gate-level NDM designs created with the Fusion Compiler or IC Compiler II tool.

Syntax

```
INDESIGN_VIRTUAL_SHIELDING: YES | NO
```

Arguments

Argument	Description
YES	Honors virtual shielding rules in an NDM design
NO (default)	Does not model virtual shielding effects

Description

Virtual shielding rules are used in the Fusion Compiler or IC Compiler II place-and-route tool to model the effect of shielding material to be added at a later step in the design process. The rules are saved in the NDM design database.

The INDESIGN_VIRTUAL_SHIELDING command directs the StarRC tool to apply virtual shielding rules during extraction. This command is primarily intended for extraction performed in the Fusion Compiler or IC Compiler II In-Design tool and is automatically specified by the set_extraction_options command within that tool.

Caution:

You can set the INDESIGN_VIRTUAL_SHIELDING command to YES in a standalone StarRC extraction run for comparison with Fusion Compiler or IC Compiler II In-Design extraction runs. However, you should not use virtual shielding rules for final signoff extraction runs because virtual shielding does not represent the final design.

The following usage notes apply to standalone StarRC runs:

- The StarRC tool models shielding material only on the same layer as the shielded net.
- Shielding is modeled on all sides of each shielded polygon.
- Nets inside skip cells are not considered.

See Also

- NDM_DATABASE

Feedback

INDUCTANCE_FREQUENCY

Specifies a frequency for mutual inductance analysis in transistor-level extraction.

Syntax

```
INDUCTANCE_FREQUENCY: freq
```

Arguments

Argument	Description
`freq`	Inductance analysis frequency, in GHz Default: 1 Allowable range: 0.001 to 100

Description

The `INDUCTANCE_FREQUENCY` command specifies the frequency used for mutual inductance analysis. This command has an effect only if the `INDUCTANCE_MODE` command is also specified in the StarRC command file.

As frequency increases, the skin depth decreases. Therefore, analysis at higher frequencies requires a finer analysis grid. The tool automatically increases the value of the `INDUCTANCE_NINC` setting as needed, which results in longer runtime.

See Also

- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_NINC
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_MIN_LENGTH

Specifies the minimum length for a resistor to be included in mutual inductance analysis performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_MIN_LENGTH: value
```

Arguments

Argument	Description
value	Minimum length of resistors (metal segments) to be included in mutual inductance analysis Units: microns Default: 1

Description

The INDUCTANCE_MIN_LENGTH command specifies the minimum length for a resistor to be included in mutual inductance analysis. Reducing the number of shapes to be included in inductance analysis improves runtime and memory performance.

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MODE
- INDUCTANCE_NINC
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_MODE

Enables mutual inductance analysis as part of a transistor-level extraction.and specifies the analysis mode.

Syntax

```
INDUCTANCE_MODE: INDUCTANCE | RELUCTANCE
```

Arguments

Argument	Description
INDUCTANCE	Presents results in terms of inductance
RELUCTANCE	Presents results in terms of reluctance
command not present	No inductance analysis

Description

By default, the StarRC tool does not analyze inductance. To enable mutual inductance analysis, include the INDUCTANCE_MODE command in the StarRC command file.

Inductance analysis is based on the FastHenry inductance field solver tool. In this mode, the tool generates an unreduced netlist that contains the physical sizes of resistors and the node locations.

For inductance analysis, the StarRC tool automatically sets the following commands:

- REDUCTION: NO
- NETLIST_NODE_SECTION: YES
- NETLIST_MERGE_SHORTED_PORTS: NO
- NETLIST_TAIL_COMMENTS: YES
- EXTRA_GEOMETRY_INFO: NODE
- PRINT_SILICON_INFO: YES

The following usage notes apply:

- The runtime and memory requirements increase with the number of inductors. For best results, limit the number of inductors to 10,000 or fewer.
- Power nets often serve as current return paths. For best inductance accuracy, set the POWER_EXTRACT command to YES. If you analyze inductance without power net extraction, the resulting inductance values tend to be overestimated.

- In cases where multiple vias exist between two conductor shapes, the analysis results in many small inductors between the vias. Set the `MERGE_VIAS_IN_ARRAY` command to `YES` to reduce the number of segments and improve analysis performance.
- Simultaneous multicorner analysis is not supported.
- This feature is mutually exclusive with clock net inductance analysis.
- Only the DSPF output netlist format is supported. Set the `NETLIST_FORMAT` command to `SPF` in the extraction command file.

Examples

The following is an example of the commands used for mutual inductance analysis:

```
INDUCTANCE_MODE: INDUCTANCE
INDUCTANCE_SELECT_BB: -36.5 12 -6.4 43.3
INDUCTANCE_SELECT_LAYER: M13 M12 M11
INDUCTANCE_REL_THRESHOLD: 1
INDUCTANCE_MIN_LENGTH: 0.7
INDUCTANCE_SELECT_NET: vo_m
INDUCTANCE_FREQUENCY: 1
POWER_EXTRACT: YES
MERGE_VIAS_IN_ARRAY: YES
```

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_NINC
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_NINC

Specifies how finely to split up metal segments for mutual inductance analysis performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_NINC: value
```

Arguments

Argument	Description
value	Number of filaments in which to divide metal segments for skin depth and proximity effect calculations Default: automatically set

Description

The INDUCTANCE_NINC command specifies the number of filaments in which to divide metal segments for skin depth and proximity effect calculations.

The StarRC tool automatically sets this parameter. Increasing this parameter manually increases accuracy at the cost of runtime and capacity. For most applications, use the default set by the tool.

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_REL_THRESHOLD

Specifies the minimum reluctance values to include in mutual inductance analysis performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_REL_THRESHOLD: value
```

Arguments

Argument	Description
value	Minimum reluctance value to include in the analysis, expressed as a percentage of the corresponding matrix diagonal value Units: per cent Default: 1

Description

If you select output in terms of reluctance by setting the INDUCTANCE_MODE command to RELUCTANCE, the tool first performs analysis in terms of inductance, then converts the results to reluctance. The INDUCTANCE_REL_THRESHOLD command improves performance by dropping relatively unimportant terms from the conversion. If you increase the value, runtime is reduced at the cost of accuracy.

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_NINC
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_S_PARAM_FREQUENCY

Specifies a frequency range for mutual inductance analysis performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_S_PARAM_FREQUENCY: (LIN | DEC) npoints start_freq stop_freq
```

Arguments

Argument	Description
LIN or DEC	Type of frequency sweep; one selection is required. Use LIN for a linear sweep and DEC for a logarithmic sweep.
npoints	Number of frequency points per decade for logarithmic sweeps; total number of frequency points for linear sweeps.
start_freq	Starting analysis frequency, in GHz Default: 1 Allowable range: 0.001 to 100
stop_freq	Ending analysis frequency, in GHz Default: 1 Allowable range: value of *start_freq* to 100

Description

The INDUCTANCE_S_PARAM_FREQUENCY command specifies the frequency range used for mutual inductance S parameter analysis. To analyze S parameters, the command file must include the INDUCTANCE_SELECT_S_BB and INDUCTANCE_S_PARAM_FREQUENCY commands in addition to the INDUCTANCE_MODE command.

For S parameter analysis, the tool considers a three-dimensional box defined by the x- and y-coordinates of the bounding box (specified in the INDUCTANCE_SELECT_S_BB command) and the layers of interest (specified in either the INDUCTANCE_SELECT_S_BB or INDUCTANCE_SELECT_LAYER command).

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_NINC

Feedback

- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_SELECT_BB

Specifies bounding box and layer information for mutual inductance analysis performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_SELECT_BB: X1 Y1 X2 Y2 [lyr1 lyr2 ...] | FILE infile
```

Arguments

Argument	Description
X1 Y1	Lower-left coordinates of the analysis bounding box Units: microns
X2 Y2	Upper-right coordinates of the analysis bounding box Units: microns
lyr1, lyr2 ...	Names of layers to consider for inductance analysis Default: all layers
infile	Name of an input file that contains bounding box and layer information

Description

Use this command to specify bounding boxes and layers to include in mutual inductance analysis.

If layer names are specified, only polygons on those layers are selected. If no layers are specified, polygons on all layers are selected.

You can provide the information in either of the following ways:

- Specify one bounding box and the layers to consider for that bounding box as arguments for the the `INDUCTANCE_SELECT_BB` command.
- Specify the name of a file that contains the information for multiple bounding boxes. Each line in the file must follow the same format as the command arguments.

If you specify layers with both the `INDUCTANCE_SELECT_LAYER` command and the `INDUCTANCE_SELECT_BB` command, only the layers that appear in both commands are selected for inductance analysis (equivalent to a logical AND operation).

Feedback

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_NINC
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_SELECT_LAYER

Specifies layers for mutual inductance analysis performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_SELECT_LAYER: db_layer1 db_layer2 ...
```

Arguments

Argument	Description
db_layer1 ...	Layers for mutual inductance analysis Default: all layers

Description

Use the INDUCTANCE_SELECT_LAYER command to specify layers to be considered in mutual inductance analysis.

If you specify layers with both the INDUCTANCE_SELECT_LAYER command and the INDUCTANCE_SELECT_BB command, only the layers that appear in both commands are selected for inductance analysis (equivalent to a logical AND operation).

The same rule applies if you use both the INDUCTANCE_SELECT_LAYER command and the INDUCTANCE_SELECT_S_BB command. However, the INDUCTANCE_SELECT_BB command and the INDUCTANCE_SELECT_S_BB command have no effect on each other because they apply to different types of analysis.

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_NINC
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_NET

Feedback

- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_SELECT_NET

Specifies nets for mutual inductance analysis performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_SELECT_NET: net1 net2 ...
```

Arguments

Argument	Description
net1 ...	Nets for mutual inductance analysis Default: all nets

Description

Use the `INDUCTANCE_SELECT_NET` command to specify the nets to be considered in mutual inductance analysis.

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_NINC
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_S_BB
- Mutual Inductance Extraction

Feedback

INDUCTANCE_SELECT_S_BB

Specifies bounding box and layer information for mutual inductance analysis of S parameters performed as part of a transistor-level extraction.

Syntax

```
INDUCTANCE_SELECT_S_BB: outfile X1 Y1 X2 Y2 [lyr1 lyr2 ...] | FILE infile
```

Arguments

Argument	Description
outfile	Name of the output file, which is created in Touchstone format
X1 Y1	Lower-left coordinates of the analysis bounding box Units: microns
X2 Y2	Upper-right coordinates of the analysis bounding box Units: microns
lyr1, lyr2 ...	Names of layers to consider for inductance analysis Default: none
infile	Name of an input file that contains bounding box and layer information

Description

Use the INDUCTANCE_SELECT_S_BB command to specify the layers and bounding box to include in mutual inductance S parameter analysis. To analyze S parameters, you must include all of the following commands in the command file: INDUCTANCE_MODE, INDUCTANCE_SELECT_S_BB, and INDUCTANCE_S_PARAM_FREQUENCY.

You can provide the information in either of the following ways:

- Specify one bounding box and the layers to consider for that bounding box as arguments for the the INDUCTANCE_SELECT_S_BB command.
- Specify the name of a file that contains the information for multiple bounding boxes. Each line in the file must follow the same format as the command arguments. The analysis generates one output file for each bounding box (each line in the file).

If you specify layers with both the INDUCTANCE_SELECT_LAYER command and the INDUCTANCE_SELECT_S_BB command, only the layers that appear in both commands are selected for inductance analysis (equivalent to a logical AND operation).

The output netlist presents the information as follows:

```
S_index n1 n2 n3 ... Gnd  NAME=S_M_index
.model S_M_index S N=num_ports TSTONEFILE=outfile
```

In this example, n1, n2, and so on are the port nodes and num_ports is the total number of ports. The ports are automatically generated by the StarRC tool. Gnd is the ground or reference node. Index is an automatically generated index number.

The output file format uses Touchstone version 2.0. For more information, see general information about this open-source format.

See Also

- INDUCTANCE_FREQUENCY
- INDUCTANCE_MIN_LENGTH
- INDUCTANCE_MODE
- INDUCTANCE_NINC
- INDUCTANCE_REL_THRESHOLD
- INDUCTANCE_S_PARAM_FREQUENCY
- INDUCTANCE_SELECT_BB
- INDUCTANCE_SELECT_LAYER
- INDUCTANCE_SELECT_NET
- Mutual Inductance Extraction

Feedback

INPUT_NAMES_ESCAPE_REMOVAL

Removes all escape characters (\) from names of the nets and instances in a LEF/DEF flow.

Syntax

```
INPUT_NAMES_ESCAPE_REMOVAL: YES | NO
```

Arguments

Argument	Description
YES (default)	Removes all escape characters from names of the net and instances.
NO	Retains all escape characters in net and instance names

Description

In the LEF/DEF flow, the INPUT_NAMES_ESCAPE_REMOVAL command removes all escape characters (\) from the net names. Because the escape character is used to make the names in the NDM flow consistent with the LEF/DEF flow, the command does not add any escape characters to the names in the NDM flow.

Examples

- When INPUT_NAMES_ESCAPE_REMOVAL : NO, the net name retains the escape characters as follows:

  ```
  u4\/n3\[1\]
  ```

- When INPUT_NAMES_ESCAPE_REMOVAL : YES, the net name removes the escape characters as follows:

  ```
  u4/n3[1]
  ```

- The following escape characters are added to indicate that the escape characters are retained when INPUT_NAMES_ESCAPE_REMOVAL command is set to NO.

  ```
  SPEF: '// DEF_ESCAPE_KEPT' is written in the header of the SPEF.
  ```

Removal of the escape characters can have the following consequences:

- A netlist file might be affected because of the change in names.
- Commands, such as FS_SELECT_NETS and NETS, might be affected.

Feedback

For example, applying the INPUT_NAMES_ESCAPE_REMOVAL changes a select net originally declared as NETS:A\/B to NETS:A/B. Because the INPUT_NAMES_ESCAPE_REMOVAL command is set to YES by default, you see the following warning message in this case:

```
INPUT_NAMES_ESCAPE_REMOVAL is enabled, please make sure that the net
 select commands remove the escape at the same time.(SX-3860)
```

Feedback

INSTANCE_PORT

Applies different parasitic resistance extraction modes to different groups of instance ports in a single extraction run.

Syntax

```
INSTANCE_PORT: CONDUCTIVE [SUFFIXED | SUFFIXED MULTIPLE | CAPACITIVE]
     | NOT_CONDUCTIVE | SUPERCONDUCTIVE | EXPLODE
     [CELL cell_name] [INST inst_name]
     [PORT port_name] [PORT_DIR IN | OUT | INOUT]
```

Arguments

Argument	Description
CONDUCTIVE (default)	Enables resistance extraction for the ports of skip cell instances. Therefore, feedthrough resistance of selected instance ports are preserved. Supplemental modes are SUFFIXED (with or without MULTIPLE) or CAPACITIVE.
NOT_CONDUCTIVE	Similar to the CONDUCTIVE MULTIPLE SUFFIXED option, but the port resistance is not written in the netlist. If the top-level material overlaps with the instance port, the conductance of the overlapping part of the top-level material is not reported.
SUPERCONDUCTIVE	When specified for a SKIP_CELLS port, the port is extracted as an ideal node with zero resistivity.
EXPLODE	Promotes selected instance port to the top level and marks it as part of the top-level net connecting it. No port nodes are generated for instance ports selected for EXPLODE extraction. Port names must be layout names. In the Calibre Connectivity Interface flow, layout text should be used instead of layout net numbers if layout text exists for the port in question.
CELL *cell_name*	Layout cell name
INST *inst_name*	Layout instance name
PORT *port_name*	Layout port name
PORT_DIR	Specifies that the extraction mode applies only to ports with the specified direction. You can specify only one direction for a single INSTANCE_PORT command. The directions are case-insensitive. **Note:** To use the PORT_DIR option in the IC Validator or CCI StarRC skip cell flow extractions, the SKIP_CELL_PORT_PROP_FILE file must have user-defined port direction settings correctly specified.

Feedback

Description

The INSTANCE_PORT command applies different parasitic resistance extraction modes to different groups of instance ports in a single extraction run. You can specify the INSTANCE_PORT command multiple times. Each use is cumulative, but if any instance ports match more than one command, the last definition overrides any previous ones.

You can apply the INSTANCE_PORT command to specific cells, instances, or ports. Specify only layout names in these arguments, not schematic names. The default for all three options is the asterisk wildcard (*).

For the CONDUCTIVE MULTIPLE and NOT_CONDUCTIVE modes, the number of port nodes is determined by the top-level resistive interaction regions. For the CONDUCTIVE mode, only one port node is created.

Instance Port Location

By default, the StarRC tool places an instance port location at the interaction point of a port and the top-level connection. If a port has multiple top-level connections, the tool places the port location at the lower-left point of the interaction with the highest metal layer.

However, when the INSTANCE_PORT command is set to CONDUCTIVE, the port resistance is added to the point-to-point resistance path, which is not desirable for interactions between instance ports and top-level fill or patch features.

For this reason, the interaction point of an instance port with top-level fill or patch polygons is not considered as a candidate for the instance port location.

You can specify an alternative instance port location closer to a driver cell by setting the INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER command to YES.

CONDUCTIVE Option

By default, one port (*|I) node per instance port is generated and no capacitance is reported for the port. The location of the node is a point of contact between the port and the top-level material. This default mode is sufficient for most applications.

Three supplemental modes can be used with INSTANCE_PORT: CONDUCTIVE, as follows:

- The CAPACITIVE option retains the capacitance of the selected instance ports.
- The SUFFIXED option modifies the port node instance name with a suffix according to the NETLIST_RENAME_PORTS command. Its effect depends on whether you also use the MULTIPLE option. The MULTIPLE option cannot be used without the SUFFIXED option.
  - If the MULTIPLE option is not used, exactly one connection point is generated, regardless of the number of interactions between the port and the top-level material. In the case of no interactions, the location of the node is random within the port. In

the case of multiple connections to the top-level material, only one is reported as a port node.

- If the MULTIPLE option is used, all of the connection points are reported as port nodes. In the case of no interactions, no port nodes are generated. In the case of a large overlap with the top-level material, multiple connection points are reported.

NOT_CONDUCTIVE Option

If the port resistors are not included in the netlist, there might be disconnected networks. However, no open warnings are issued, because the net is known to be connected by feedthrough. Port nodes for a given instance port are generated at every top-level interaction point. If there is more than one interaction point, multiple port nodes are included in the netlist. In the case of no interaction with top-level material, no port node is generated.

SUPERCONDUCTIVE Option

The StarRC tool assumes ideal (zero) resistivity when extracting port shapes inside the skip cell. The port shapes act as shorts and no resistance is extracted for the port shapes.

EXPLODE Option

The StarRC tool promotes a selected instance port to the top level and marks it as part of the top-level net that connects to it. Port nodes are not created for instance ports that are set to EXPLODE, and *I or *|I statements are not generated.

Examples

- The following command sets all instance ports to CONDUCTIVE:

```
INSTANCE_PORT: CONDUCTIVE
```

- The following example sets all ports in cell ANTENNA to EXPLODE (to the top level), leaving all other instance ports set to CONDUCTIVE:

```
INSTANCE_PORT: CONDUCTIVE
INSTANCE_PORT: EXPLODE CELL ANTENNA
```

- The following example sets ports VDD and VSS in all cells with suffixes, multiple times if there is more than one connection point. Ports VSS and VDD in cell ANTENNA are retained, but all other ports in cell ANTENNA are exploded.

```
INSTANCE_PORT: CONDUCTIVE
INSTANCE_PORT: EXPLODE CELL ANTENNA
INSTANCE_PORT: CONDUCTIVE MULTIPLE SUFFIXED CELL * PORT VDD VSS
```

Feedback

- The following example sets all instance ports to `CONDUCTIVE SUFFIXED`, except for instance ports with cell names that begin with A, which are set to `NOT_CONDUCTIVE`:

```
INSTANCE_PORT: CONDUCTIVE SUFFIXED
INSTANCE_PORT: NOT_CONDUCTIVE CELL A*
```

- The following example sets instance ports that match CELL B* and PORT VDD* to `CONDUCTIVE`, while all other ports remain set to `NOT_CONDUCTIVE`:

```
INSTANCE_PORT: NOT_CONDUCTIVE
INSTANCE_PORT: CONDUCTIVE SUFFIXED CELL B* !AB* PORT VDD*
```

- The following example sets all output ports to be superconductive:

```
INSTANCE_PORT: SUPERCONDUCTIVE PORT_DIR OUT
```

- The following example sets all input ports of cell A to be superconductive:

```
INSTANCE_PORT: SUPERCONDUCTIVE CELL A PORT_DIR IN
```

- The following example sets only the input ports of cell A to be superconductive, because only the first `PORT_DIR` keyword is recognized in a single invocation of the `INSTANCE_PORT` command.

```
INSTANCE_PORT: SUPERCONDUCTIVE CELL A PORT_DIR IN PORT_DIR OUT
```

- The following example sets all input and output ports of cell A to be superconductive:

```
INSTANCE_PORT: SUPERCONDUCTIVE CELL A PORT_DIR IN
INSTANCE_PORT: SUPERCONDUCTIVE CELL A PORT_DIR OUT
```

See Also

- INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER
- INSTANCE_PORT_REPORT_FILE
- INSTANCE_PORT_OPEN_CONDUCTANCE
- NETLIST_RENAME_PORTS
- SKIP_CELLS
- SKIP_CELL_PORT_PROP_FILE
- SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR
- TRANSLATE_RETAIN_BULK_LAYERS

Feedback

INSTANCE_PORT_REPORT_FILE

Writes the instance port types.

Syntax

```
INSTANCE_PORT_REPORT_FILE: file_name
```

Arguments

Argument	Description
file_name	File name Default: none

Description

The INSTANCE_PORT_REPORT_FILE command reports the instance port types that are not set to CONDUCTIVE with the INSTANCE_PORT command. The information of instance port types are reported in the following format:

```
Port <port_name> of instance <instance_name> is treated as
 <instance_port_type>
```

Examples

The following command writes information of all instance ports to the port.txt file that are not set to the CONDUCTIVE type with the INSTANCE_PORT command:

```
INSTANCE_PORT_REPORT_FILE: port.txt

# contents in the port.txt file
Port 'o' of instance 'PLACE_SUBXYX_1139' is treated as: SUPERCONDUCTIVE
```

See Also

- INSTANCE_PORT
- INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER
- SKIP_CELL_PORT_PROP_FILE
- SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR

Feedback

INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER

Specifies how to determine instance port locations. Valid only for gate-level flows.

Syntax

```
INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER: YES | NO
```

Arguments

Argument	Description
YES	Places instance ports close to drivers
NO (default)	Uses default instance port location

Description

The INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER command specifies how to determine an instance port location when the instance port has multiple top-level connections.

For instance ports with single connections, StarRC tool sets the instance port (*I) location at the interaction point of the port and the top-level connection.

For instance ports with multiple top-level connections, the tool places the instance port by default at the lower-left coordinate of the interaction point with the highest metal layer. You can alternatively choose to place the instance port at the location with the smallest point-to-point resistance by setting the INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER command to YES.

Note:

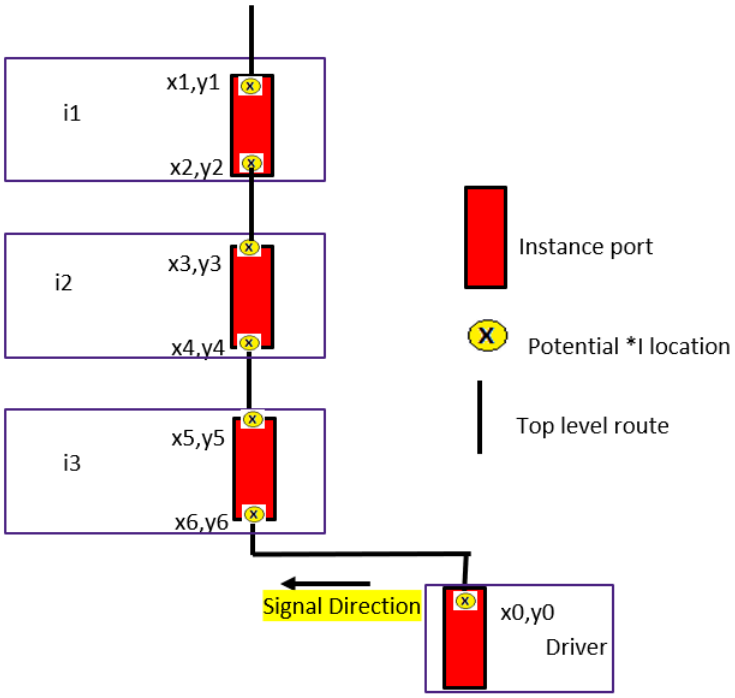
To use the INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER command in the IC Validator or CCI StarRC skip cell flow extractions, the SKIP_CELL_PORT_PROP_FILE file must have user-defined port direction settings correctly specified.

Examples

For example, in Figure 201 the instance ports of skip cells i1, i2, and i3 are feedthrough nets used as routing resources. The driver cell instance port is at location (x0,y0). The potential skip cell instance port locations are marked with the label X. By default, the tool selects the instance port locations at coordinates (x2,y2), (x4,y4), and (x6,y6) because these are the lowest and leftmost points of the interaction of the skip cells with the top-level routing.

Even if the INSTANCE_PORT: CONDUCTIVE command is used, the resistance of the instance port inside the skip cells is not added to the point-to-point resistance.

Figure 201 Instance Port Example 1



However, in Figure 202 the driver cell is in a different location with respect to the skip cells.

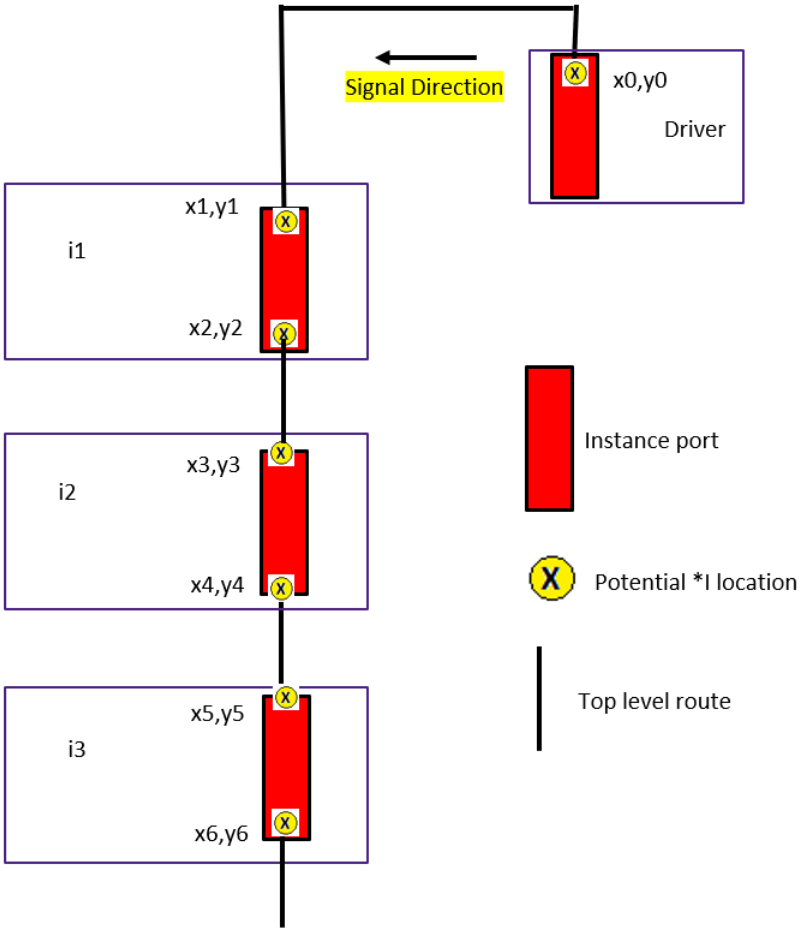
The instance port is still at location (x0,y0). As in the previous example, by default, the tool selects skip cell instance port locations at coordinates (x2,y2), (x4,y4), and (x6,y6) because these are the lowest and leftmost points of the interaction of the skip cells with the top-level routing.

In this case, if the `INSTANCE_PORT: CONDUCTIVE` command is used, the resistance of the instance port inside the skip cells is added to the point-to-point resistance, which leads to double counting of the instance port resistances.

Set the `INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER` command to `YES` to use the lowest point-to-point resistance to select instance port locations. In Figure 202, the instance port locations are selected to be (x1,y1), (x3,y3), and (x5,y5).

Feedback

Figure 202 Instance Port Example 2



See Also

- INSTANCE_PORT
- INSTANCE_PORT_REPORT_FILE
- SKIP_CELL_PORT_PROP_FILE

Feedback

INSTANCE_PORT_OPEN_CONDUCTANCE

Defines a fixed conductance value for shorting resistors that connect port members inside a skip cell.

Syntax

```
INSTANCE_PORT_OPEN_CONDUCTANCE: cond_value
```

Arguments

Argument	Description
cond_value	A fixed conductance value for shorting resistors used to connect members of a port that are not electrically connected inside the skip cell Units: mho Default: 10.0 (equivalent to 0.1 ohm)

Description

The INSTANCE_PORT_OPEN_CONDUCTANCE command defines a fixed conductance value for shorting resistors used to connect members of a port that are not resistively connected inside a defined skip cell. This command affects only conductive instance ports.

This might happen when the instance port material is not completely ported. Often, small routing targets at the edges, instead of the entire port, are used as the port definitions. In this case, the StarRC tool inserts resistors with the specified conductance value to prevent opens in the output netlist. If you are also using the NETLIST_TAIL_COMMENTS command, the resistor width is reported as 9 microns to facilitate identification.

The tool does not insert shorting resistors between ports (or members of a port) and the substrate.

See Also

- INSTANCE_PORT
- NETLIST_TAIL_COMMENTS
- TRANSLATE_RETAIN_BULK_LAYERS

Feedback

INSTANCE_TYPE

Specifies whether layout or schematic instance names are used for instance selection.

Syntax

```
INSTANCE_TYPE: LAYOUT | SCHEMATIC
```

Arguments

Argument	Description
`LAYOUT` (default)	Uses layout instance names
`SCHEMATIC`	Uses schematic instance names matched during LVS processing

Description

The `INSTANCE_TYPE` command specifies whether layout or schematic cell names are used for instances specified to be skipped in the `SKIP_INSTANCES` command.

This command is ignored if `XREF: NO` is specified.

Examples

```
INSTANCE_TYPE: LAYOUT
```

See Also

- SKIP_INSTANCES
- SKIP_CELLS

Feedback

ITF_FILE

Specifies a file containing ITF commands for the direct ITF flow. Valid only for transistor-level flows.

Syntax

```
ITF_FILE: file_name
```

Arguments

Argument	Description
file_name	The ITF file name Default: none

Description

This command allows you to experiment with process changes by specifying an ITF file directly in the StarRC command file.

The direct ITF flow does not use the grdgenxo program and does not generate or use an nxtgrd file. This command is supported only for capacitance extraction in transistor-level flows.

See Also

- The Direct ITF Flow

Feedback

KEEP_SUBCONT_MODELS

Specifies a list of models for which to keep all substrate contacts. Valid only for transistor-level flows.

Syntax

```
KEEP_SUBCONT_MODELS: model_list
```

Arguments

Argument	Description
model_list	Model list Default: !*

Description

The KEEP_SUBCONT_MODELS command allows you to specify how the StarRC tool handles substrate contacts during transistor-level extraction. This command has an effect only when the TRANSLATE_RETAIN_BULK_LAYERS command is set to CONLY.

By default, the tool retains only one substrate contact (subCont instance) for each substrate polygon. However, you can keep all substrate contacts for specific features by including the KEEP_SUBCONT_MODELS command and specifying the models of interest.

See Also

- TRANSLATE_RETAIN_BULK_LAYERS

Feedback

KEEP_VIA_NODES

Defines via nodes as the original nodes that a via resistor connects.

Syntax

```
KEEP_VIA_NODES: YES | NO | MAPPING_FILE
```

Arguments

Argument	Description
YES	Preserves original nodes that a via resistor connects
NO (default)	Does not preserve original nodes
MAPPING_FILE	Follows via reduction instructions in the mapping file

Description

The KEEP_VIA_NODES command defines via nodes as the original nodes that a via resistor connects. The original nodes might be reduced or merged with other nodes, depending on the conductor configuration and reduction mode. Setting the KEEP_VIA_NODES command to YES preserves the original nodes.

You can specify how reduction is applied to vias by setting the KEEP_VIA_NODES command to MAPPING_FILE and including the REDUCE_VIA option in the via_layers section in the mapping file. The reduction options are as follows:

- yes (reduce vias)
- no (do not reduce vias)
- power (reduce vias only if they belong to power nets)
- signal (reduce vias only if they belong to signal nets)

See Also

- REDUCTION

Feedback

LEF_FILE

Creates a white-space-delimited list of LEF format files containing complete library cell descriptions.

Syntax

```
LEF_FILE: technology_lef library_lef
LEF_FILE: library_lef macro_lef
LEF_FILE: macro_lef custom_lef
```

Arguments

Argument	Description
technology_lef	LEF file that contains the technology information
library_lef	LEF file that contains the cell library information
macro_lef	LEF file that contains the core cell information
custom_lef	LEF file that contains the custom cell and block information

Description

This command, which is mandatory for LEF/DEF flows, can be specified multiple times in a single command file. The order in which the LEF files are specified is very important.

There are three types of LEF files: the technology LEF file, the standard cell LEF file, and the macro LEF file. The technology LEF file must be listed first in the command file. Every layer defined in the technology LEF file must be mapped to an nxtgrd file layer by a mapping file entry. Undefined layers that in a LEF file cause the StarRC tool to issue an error and exit.

If subsequently specified LEF files contain the following technology information, the StarRC tool reads this information and uses it to override previously read values:

- VIA constructs
- WIDTH, DIRECTION, and WIREEXTENSION attributes under a LAYER construct

The tool ignores all other technology information in LEF files that are read after the first technology LEF file.

The standard cell LEF file and the macro LEF file can be listed in any order after the technology LEF file. You do not need to provide LEF descriptions for DEF macros specified in a `MACRO_DEF_FILE` command. Either a LEF or DEF description is required for every member of the list specified by the `SKIP_CELLS` command.

Feedback

See Also

- GDS_LAYER_MAP_FILE
- MACRO_DEF_FILE
- MAPPING_FILE
- SKIP_CELLS

Feedback

LEF_SPACING_OBS

Specifies how to treat OBS shapes (blockage regions) in a LEF/DEF design. Valid for gate-level flows.

Syntax

```
LEF_SPACING_OBS: NO | YES
```

Arguments

Argument	Description
YES	Translates OBS objects with SPACING values and considers the setting of the LEF_ZERO_SPACING_OBS command
NO (default)	Does not translate OBS objects with SPACING keywords, regardless of the spacing value

Description

The LEF_SPACING_OBS command specifies how the StarRC tool translates OBS shapes with SPACING values in a LEF/DEF design. When you set the LEF_SPACING_OBS command to the following values,

- YES: The StarRC tool treats an OBS shape as a single polygon. If the SPACING value of the OBS shape is zero or 0, the LEF_ZERO_SPACING_OBS command controls whether the OBS region is translated.
- NO (the default): The StarRC tool ignores all OBS shapes with a SPACING value, regardless of the actual SPACING value.

Note:

Do not use the LEF_USE_OBS, LEF_SPACING_OBS, and LEF_ZERO_SPACING_OBS commands in signoff flows, because blockages do not represent the final design.

See Also

- LEF_USE_OBS
- LEF_ZERO_SPACING_OBS

Feedback

LEF_USE_OBS

Enables the translation of OBS shapes (routing blockages) in a LEF/DEF design. Valid for gate-level flows.

Syntax

```
LEF_USE_OBS: YES | NO | SHRINK
```

Arguments

Argument	Description
YES (default)	Translates OBS objects as real metal shapes and allows coupling to these shapes
NO	Does not translate OBS objects
SHRINK	Translates OBS objects with shrink spacing as real metal shapes and allows coupling to these shapes

Description

Many LEF file macro definitions contain groups of shapes under the heading OBS (obstruction). These are blockage layers that restrict the top-level router and typically are a close approximation of the actual cell layout.

By default, the StarRC tool translates OBS shapes. Set the LEF_USE_OBS command to NO to ignore the OBS shapes in all skip cells.

The LEF_USE_OBS command has no effect on macro definitions for which supplemental GDSII information is provided with the GDS_FILE command.

See Also

- GDS_FILE
- SKIP_CELLS
- LEF_SPACING_OBS
- LEF_ZERO_SPACING_OBS

Feedback

LEF_ZERO_SPACING_OBS

Specifies whether to translate zero spacing OBS shapes (routing regions) in a LEF/DEF design.

Syntax

```
LEF_ZERO_SPACING_OBS: NO | YES
```

Arguments

Argument	Description
YES	Translates OBS shapes with a SPACING value of 0
NO (default)	Does not translate OBS shapes with a SPACING value of 0

Description

The LEF_ZERO_SPACING_OBS command specifies whether the StarRC tool should translate zero spacing OBS shapes in a LEF/DEF design. The LEF_ZERO_SPACING_OBS command is effective only if the LEF_USE_OBS and LEF_SPACING_OBS commands are set to YES

When you set the LEF_ZERO_SPACING_OBS command to the following values,

- YES: The StarRC tool translates zero-spacing OBS shapes.
- NO (the default): The StarRC tool ignores any zero-spacing OBS shapes.

Note:

Do not use the LEF_USE_OBS, LEF_SPACING_OBS, and LEF_ZERO_SPACING_OBS commands in signoff flows, because blockages do not represent the final design.

See Also

- LEF_USE_OBS
- LEF_SPACING_OBS

Feedback

LPE_DEVICES

Specifies the device model names that follow the LPE_PARAM rules.

Syntax

```
LPE_DEVICES: param_name device_model_1 [device_model_2 …]
```

Arguments

Argument	Description
param_name	A user-defined LPE parameter name. The specified parameter name should match the parameter names specified by the LPE_PARAM command.
device_model	Device models to which the LPE parameter should be applied

Description

The LPE_DEVICES specifies the device model names that follow the LPE_PARAM rules. This command must be used in conjunction with the LPE_PARAM command.

If the list of device models is very long, you can use multiple LPE_DEVICES commands for the same parameter. The lists of device models are concatenated.

Examples

```
LPE_DEVICES: pre_layout nfet pfet
LPE_DEVICES: nores ncap
```

See Also

- LPE_PARAM
- LPE_FLAGS_SETTING

Feedback

LPE_FLAGS_SETTING

Specifies how to apply parameters specified in the LPE_PARAM command.

Syntax

```
LPE_FLAGS_SETTING: NETLIST | EXTRACTION
```

Arguments

Argument	Description
NETLIST (default)	Applies parameters based on the extraction status of nets connected to terminals of user-defined instances as in the final netlist
EXTRACTION	Applies parameters on a global level based on the EXTRACTION command setting

Description

The LPE_DEVICES and LPE_PARAM commands allow you to control which LPE parameters are used for user-defined instances. Specifying an appropriate LPE parameter for each user-defined instance results in the best accuracy for overall simulation. The default setting NETLIST for the LPE_FLAGS_SETTING command assumes this approach.

Alternatively, you can simplify the application of LPE parameters by setting the LPE_FLAGS_SETTING command to EXTRACTION. In this case, the extraction mode set in the EXTRACTION command (R, C, or RC) is used for all LPE parameters of user-defined instances. This setting tends to produce pessimistic results.

Examples

In this example, even though the LPE_PARAM command contains flags for all modes for device type nfet, the LPE_FLAGS_SETTING command forces the use of the capacitance-mode flag (as set in the EXTRACTION command) for all user-defined instances.

```
EXTRACTION: C
LPE_DEVICES: pre_layout nfet pfet
LPE_FLAGS_SETTING: EXTRACTION
LPE_PARAM: nfet NORC 1 C 3 R 2 RC 0 PINEXCEPT 4
```

See Also

- LPE_PARAM
- LPE_DEVICES
- EXTRACTION

Feedback

LPE_PARAM

Defines a netlist parameter for user-defined instances.

Syntax

```
LPE_PARAM: param_name mode1 value1 [mode2 value2…]
           [PINEXCEPT pinIds]
           [PINOPERATION AND | OR]
```

Arguments

Argument	Description
param_name	The name of the LPE parameter
mode	The extraction mode used to netlist the device. Valid modes are R, C, RC, and NORC.
value	Corresponds to flags in the device simulation SPICE model. The value can be customized based on flows.
PINEXCEPT *pin_IDs*	The list of pin indexes to be ignored while computing the extraction mode for the device. Pin indexes start at 0 for the first pin of a device.
PINOPERATION AND | OR (default is AND)	The method used to determine the value of the gate-to-contact coupling LPE parameter. When you set PINOPERATION to OR, the tool uses an OR operation to evaluate the extraction mode of the terminals.

Description

The LPE_PARAM command defines a netlist parameter for user-defined instances. This post-layout parameter is set based on the extraction mode of the nets connected to the user-defined instance. You can optionally use the LPE_FLAGS_SETTING and EXTRACTION commands to specify one extraction mode to use for all instances.

The command can be used to control, based on a user-defined parameter and value, which parasitics are taken from simulation SPICE models and which parasitics are extracted by the StarRC tool.

This command must be used in conjunction with the LPE_DEVICES command.

The PINEXCEPT option causes the list of pin indexes to be ignored while computing the extraction mode for the device. For example, it could be used to ignore the bulk in a MOS device (PINEXCEPT 3), so that only the net connected to drain, source and gate would be taken into account for computing an LPE parameter value.

There can be only one LPE_PARAM definition for each parameter. If multiple LPE_PARAM definitions exist for a single parameter, then the last definition overrides the previous definitions.

For completeness, all extraction modes should be specified in the LPE_PARAM command. However, if the extraction mode is not specified in the LPE_PARAM command, the tool tries to compute the value.

If RC is not described for a parameter, then capacitance extraction has no impact on its value. If capacitance extraction is performed, the tool must use the NOR value, which is the default.

Examples

The following defines three parameters for a user-defined instance along with the extraction mode and the corresponding flag setting in the SPICE model.

```
LPE_PARAM: pre_layout_local NORC 1 C 3 R 2 RC 0 PINEXCEPT 4
LPE_PARAM: setres NORC -2 C -2 R 0 RC 0
LPE_PARAM: setcap NORC -1 C 0 R -1 RC 0
```

The following table lists four user-defined instances and the extraction mode setting for pin A and pin B of each instance. The table shows the value of the LPE parameter when PINOPERATION is set to an OR operation.

Pin A	Pin B	AND Operation	OR Operation
RC	RC	RC	RC
	R	R	RC
	C	C	RC
	NORC	NORC	RC
R	RC	R	RC
	R	R	R
	C	NORC	RC
	NORC	NORC	R
C	RC	C	RC
	R	NORC	RC
	C	C	C
	NORC	NORC	C

Feedback

Pin A	Pin B	AND Operation	OR Operation
NORC	RC	NORC	RC
	R	NORC	R
	C	NORC	C
	NORC	NORC	NORC

See Also

- LPE_DEVICES
- LPE_FLAGS_SETTING

Feedback

LVS_EXTRACTION_REPORT_FILE

Specifies the Calibre

Syntax

```
LVS_EXTRACTION_REPORT_FILE: lvs_rpt
```

Arguments

Argument	Description
lvs_rpt	Calibre extraction report file name Default: none

Description

If the Calibre query file uses the `LVS SETTINGS REPORT WRITE` command to write an extraction report, the StarRC `CALIBRE_QUERY_FILE` command automatically obtains the runset file and PDBA information from the query file. The `CALIBRE_RUNSET` and `CALIBRE_PDBA_FILE` commands are invalid in this case.

Instead of reading the automatically determined extraction report, you can specify a different Calibre extraction report by using the `LVS_EXTRACTION_REPORT_FILE` command in the StarRC command file. The file specified by the StarRC `LVS_EXTRACTION_REPORT_FILE` command takes precedence over the file named in the Calibre `LVS SETTINGS REPORT WRITE` command.

Examples

```
LVS_EXTRACTION_REPORT_FILE: cci_rpt
```

See Also

- CALIBRE_PDBA_FILE
- CALIBRE_QUERY_FILE
- CALIBRE_RUNSET
- MILKYWAY_DATABASE

Feedback

MACRO_DEF_FILE

Specifies a list of DEF files that define macros referenced in the file specified by the TOP_DEF_FILE command.

Syntax

```
MACRO_DEF_FILE: macro_def_file1 macro_def_file2 …
```

Arguments

Argument	Description
macro_def_file	Macro DEF files referenced in the DEF file specified by the TOP_DEF_FILE command

Description

The MACRO_DEF_FILE command is a list of DEF files that define macros referenced in the file specified by the TOP_DEF_FILE command. There is no need to provide LEF descriptions with the LEF file for macros on this list because the DEF descriptions are used instead.

This command can be specified multiple times in a StarRC command file. The order of the files in the list is not important.

The cells described in a file specified by the MACRO_DEF_FILE command can also be in the list of skip cells in the file specified by the SKIP_CELLS command. For example, you might extract these macro cells to produce a full-chip flat netlist, or you might skip them if you are performing hierarchical extraction.

The MACRO_DEF_FILE command accepts gzip-compressed DEF files.

See Also

- LEF_FILE
- SKIP_CELLS
- TOP_DEF_FILE

Feedback

MAGNIFICATION_FACTOR

Scales input data uniformly for all layers.

Syntax

```
MAGNIFICATION_FACTOR: value
```

Arguments

Argument	Description
value	Scaling factor; a positive floating-point number Default: 1.0

Description

The MAGNIFICATION_FACTOR command performs scaling of layout dimensions for all layers. Scaling does not affect the connectivity of the layout network.

Specifying a value less than 1 indicates a shrink; a value greater than 1 indicates expansion. The scaling operation is performed on a database only while reading it.

The nxtgrd file must contain rules for minimum width, spacing, and tables associated with the shrunk or enlarged database.

You cannot use the MAGNIFICATION_FACTOR command with the HALF_NODE_SCALE_FACTOR ITF command.

See Also

- HALF_NODE_SCALE_FACTOR
- MAGNIFY_DEVICE_PARAMS

Feedback

MAGNIFY_DEVICE_PARAMS

Controls the behavior of the MAGNIFICATION_FACTOR command.

Syntax

```
MAGNIFY_DEVICE_PARAMS: YES | NO
```

Arguments

Argument	Description
YES (default)	Applies the value specified in the MAGNIFICATION_FACTOR command to SPICE device parameters
NO	Does not apply the value of the MAGNIFICATION_FACTOR to the SPICE parameters

Description

The MAGNIFY_DEVICE_PARAMS command controls the behavior of the MAGNIFICATION_FACTOR command. When you specify MAGNIFY_DEVICE_PARAMS: YES, all designed device parameters in the SPICE netlist are multiplied by the factor set by the MAGNIFICATION_FACTOR command. Table 69 lists the parameters affected by the MAGNIFY_DEVICE_PARAMS command.

Table 69 Device Parameters Affected By MAGNIFY_DEVICE_PARAMS

Device type	Magnified parameters
Diode	area, pj
Resistor	l, w
Capacitor	l, w
MOS	ad, as, pd, l, w, sa ,sb, and sc
BJT	area

See Also

- MAGNIFICATION_FACTOR

Feedback

MAPPING_FILE

Specifies the file containing physical layer mapping information between the input database and the specified nxtgrd file.

Syntax

```
MAPPING_FILE: file_name
```

Arguments

Argument	Description
file_name	The mapping file name Default: none

Description

This command is required for all flows. Every connected layer must be mapped.

You can elect to override sheet resistance values specified in the nxtgrd file by specifying new values in the mapping file specified by the `MAPPING_FILE` command.

For simultaneous multicorner flows, you can specify a mapping file for each corner by including mapping file commands in the corner definitions in the corners file. If a corner definition does not include a `MAPPING_FILE` command, a global mapping file must be defined in the top-level command file. If every corner includes a mapping file, a global mapping file is not necessary; if one exists, it is ignored.

All mapping files for simultaneous multicorner flows must be consistent in terms of the number of database and ITF file layers and their mapping relationships. Each mapping file category, such as conductors or vias, should also be consistent. The only allowed variations are the RPSQ value for conductors and the RPV and AREA values for vias.

For more information about mapping files, see Chapter 16, Mapping Files.

See Also

- TCAD_GRD_FILE
- CORNERS_FILE
- SIMULTANEOUS_MULTI_CORNER

Feedback

MARKER_GENERATION

Specifies how to generate pin shapes for a database.

Syntax

```
MARKER_GENERATION: AUTOMATIC | USER
```

Arguments

Argument	Description
`AUTOMATIC` (default)	The tool generates pin shapes based on connectivity
`USER`	The user generates the pin shapes. Not valid for the Calibre Connectivity Interface flow.

Description

The StarRC tool requires pin shapes to define the instance ports. Pin shapes are also referred to as markers.

There are two ways to create a pin shape. The first is to identify a special output layer that generates the pin shape, and the second is to have the tool automatically generate pin shapes, based on connectivity.

For most purposes, the `AUTOMATIC` setting is preferred because it is more robust. The `USER` setting allows more flexibility but requires great care when creating marker shapes and a rigorous knowledge of the routing methodology to avoid creating opens and shorts in the extraction. The `USER` setting is not valid for the Calibre Connectivity Interface flow.

When you specify the `MARKER_GENERATION:USER` command, you must

- Use one of the following techniques to generate a pin shape: text-based markers, ID-layer markers, or connection-based markers.
- Specify marker layers in the mapping file with the `marker_layers` statement.

See Also

- MAPPING_FILE

Feedback

MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER

Specifies the maximum number of virtual via segments in a trench contact process.

Syntax

```
MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER: number_of_segments
```

Arguments

Argument	Description
number_of_segments	Maximum number of via segments; an integer Units: none Default: 1000000

Description

Trench contacts can have tall covertical layers that are not connected by physical vias. To extract vertical resistance, virtual vias are inserted between the trench contact conductors. The StarRC tool segments these vias automatically to create a distributed resistance network.

The `MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER` command specifies the maximum number of virtual via segments in a trench contact process.

Errors

You can use the `TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO` command without the `MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER` command. However, if you use the `MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER` command alone, the StarRC tool issues an error message.

Examples

```
MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER: 4
```

See Also

- TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO

Feedback

MERGE_INSTANCE_PORTS

Decreases the effective resistance connecting the schematic port to the rest of the net when set to YES.

Syntax

```
MERGE_INSTANCE_PORTS: YES | NO
```

Arguments

Argument	Description
YES	Electrically merges all ports, which causes the branches to be treated as parallel-connected resistive paths during extraction, netlist reduction, and netlist generation
NO (default)	Matches one of the layout ports to the single schematic port, leaving the other branches as dangling in the parasitic netlist

Description

This command is valid only when the XREF:COMPLETE command is also used.

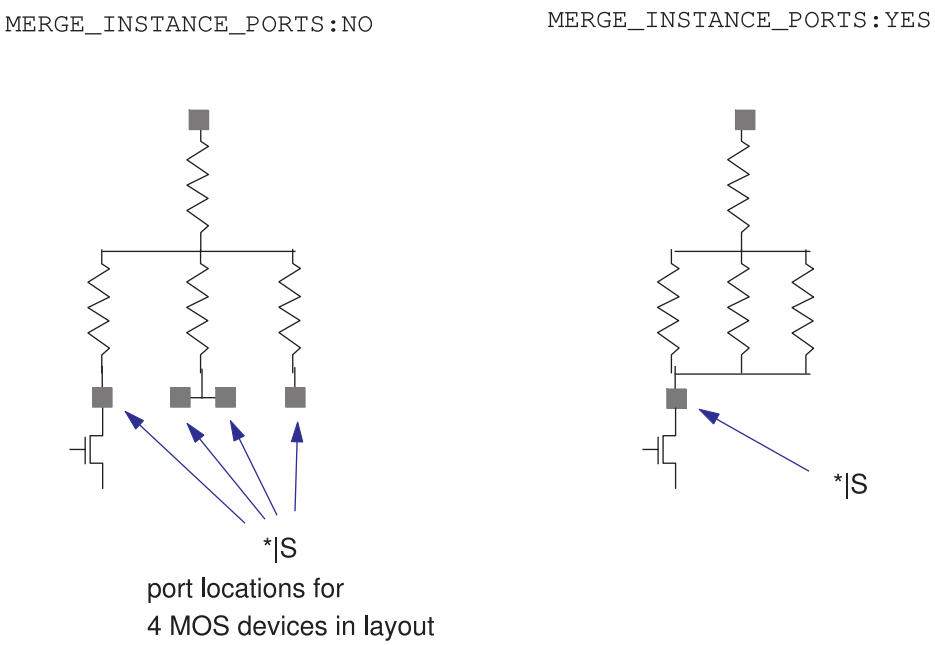
For parallel 1:N (schematic:layout) merging, the command electrically merges all of the N layout instance ports into a single port for netlist generation on each of the source, drain, and gate nets.

For parallel M:N merging where N > M, the command electrically merges each of the extra unmatched (N-M) layout ports with one of the matched M layout ports. The final parasitic netlist contains M ports for each of the source, drain, and gate nets.

Figure 203 illustrates the effect of the command.

Feedback

Figure 203 XREF:COMPLETE Parasitic Netlist With MERGE_INSTANCE_PORTS



See Also

- XREF :COMPLETE

Feedback

MERGE_PARALLEL_DEVICES

Enables merging of parallel devices identified by the IC Validator tool.

Syntax

```
MERGE_PARALLEL_DEVICES: YES  |  NO
```

Arguments

Argument	Description
YES	Enables parallel device merging
NO (default)	Disables parallel device merging

Description

The MERGE_PARALLEL_DEVICES command enables the StarRC tool to use information provided by the IC Validator tool about merged parallel devices. This capability provides netlist reduction for the instance section of the output netlist. The command is valid for the IC Validator flow only.

During layout versus schematic checking, the IC Validator tool identifies parallel layout devices based on option settings in the tool. You must verify the correct use of the IC Validator options. Merging is typically used for devices with identical properties. The StarRC tool uses the IC Validator binary cdb file directly and does not check the validity of the merging operation.

When parallel device merging occurs, one device instance (known as the primary instance) from a set of parallel devices is selected to represent the set. The device without the at sign (@) delimiter is retained as the primary instance. The number of merged devices is added to the device properties using the format m=N, where N is the number of merged devices.

Merging parallel devices might not preserve resistances that exist on the paths between the parallel devices.

The MERGE_PARALLEL_DEVICES command has an effect only if the netlist instance section is enabled with the NETLIST_INSTANCE_SECTION command and the XREF command is set to YES.

For example, consider parallel (and identical) device instances as follows:

```
XA/T2 A/T2:DRN A/T2:GATE A/T2:SRC A/T2:BULK pmos l=0.02u
ps=4.8e-07 as=1.19e-14 pd=2.3e-07 ad=5.1e-15 w=0.3u
XA/T2@2 A/T2@2:DRN A/T2@2:GATE A/T2@2:SRC A/T2:BULK pmos l=0.02u
ps=4.8e-07 as=1.19e-14 pd=2.3e-07 ad=5.1e-15 w=0.3u
```

After merging, only one instance (the master instance) is reported, with its original properties and the additional `m` property set to 2, indicating that two of these devices appear in parallel in the layout:

```
XA/T2 A/T2:DRN A/T2:GATE A/T2:SRC A/T2:BULK pmos l=0.02u
ps=4.8e-07 as=1.19e-14 pd=2.3e-07 ad=5.1e-15 w=0.3u m=2
```

Feedback

MERGE_VIAS_IN_ARRAY

Specifies whether vias in an array are merged.

Syntax

```
MERGE_VIAS_IN_ARRAY: YES | NO
```

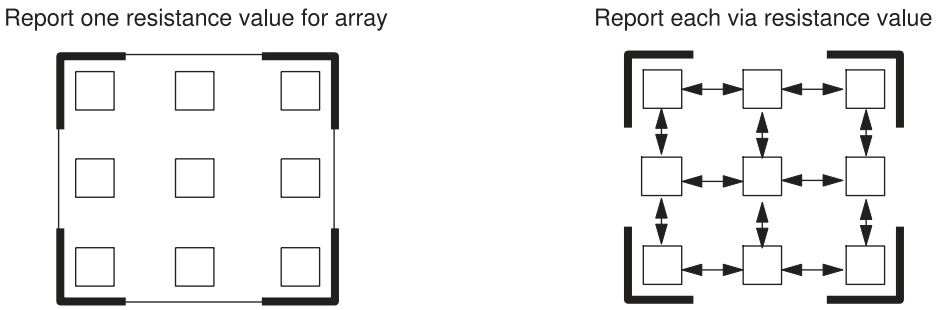
Arguments

Argument	Description
YES (default for gate-level extraction)	Merges resistance values for a via array
NO (default for transistor-level extraction)	Reports parasitic resistors for each individual via and parasitic resistors between vias

Description

The StarRC tool can merge via arrays, as illustrated in Figure 204.

Figure 204 Effect of Via Merging on Resistances



The StarRC tool handles via merging differently for gate-level and transistor-level extraction, as follows:

- Gate-level flows

  Vias are merged by default and one resistance is reported for the array. To disable via merging, set the MERGE_VIAS_IN_ARRAY command to NO.

- Transistor-level flows

  The effect of the MERGE_VIAS_IN_ARRAY command is shown in Table 70. The MAX_VIA_ARRAY_LENGTH and MAX_VIA_ARRAY_SPACING parameters in the via_layers section of the mapping file take precedence over the MERGE_VIAS_IN_ARRAY

Feedback

command. If the mapping file parameters are set, via merging takes place. If the mapping file parameters are not set, via merging occurs only if the `MERGE_VIAS_IN_ARRAY` command is set to `YES`.

The final number of vias after merging can be more than one. Transistor-level via merging applies to both power and signal nets, but not to contacts or virtual vias.

Table 70 Via Merging Behavior For Transistor-Level Flows

MERGE_VIAS_IN_ARRAY	Parameters are set in mapping file	Parameters are not set in mapping file
YES	Merge via array based on mapping file parameters	Merge via array to an extent calculated by internal heuristics
NO (or not set)	Merge via array based on mapping file parameters	Do not merge vias

Note:

In StarRC versions earlier than N-2017.12, via merging in transistor-level flows does not occur if the parameters are not set in the `via_layers` section of the mapping file.

Examples

Example 1

```
MERGE_VIAS_IN_ARRAY:YES
```

This command results in the following output:

```
13 abc[4]:45 abc[4]:46 0.72000// $a=2.00000 $lvl=5
```

Example 2

```
MERGE_VIAS_IN_ARRAY:NO
VIA_COVERAGE_OPTION_FILE: via1
```

These commands result in the following output:

```
14 abc[4]:45 abc[4]:46 1.44000 // $a=1.00000 $lvl=5
15 abc[4]:54 abc[4]:55 1.44000 // $a=1.00000 $lvl=5
```

See Also

- KEEP_VIA_NODES

Feedback

MESH_RESISTANCE_EXTRACTION_LAYERS

Specifies layers in which to perform mesh resistance extraction. Valid only for transistor-level flows.

Syntax

```
MESH_RESISTANCE_EXTRACTION_LAYERS: db_layer1 db_layer2 ...
```

Arguments

Argument	Description
db_layer 1 ...	Physical database layers for mesh resistance extraction Default: none

Description

Use the `MESH_RESISTANCE_EXTRACTION_LAYERS` command to specify layers for mesh resistance extraction.

This command requires an Ultra license.

The following usage notes apply:

- The specified layers must be physical database layers, not ITF layers.
- All resistances in the specified layers are extracted using mesh resistance analysis.
- All layers for mesh resistance analysis must be specified in the same command.
- If the command file contains multiple `MESH_RESISTANCE_EXTRACTION_LAYERS` commands, the last command takes precedence and the earlier commands are ignored.
- If a specified layer does not exist, the tool issues an SX-3771 warning message.

Mesh resistance analysis might require longer runtime than standard resistance analysis. In addition, the netlist size increases due to the larger number of resistance nodes, which also affects subsequent simulation runtime. Use the `REDUCTION: HIGH` command to mitigate these effects.

Feedback

MESSAGE_SUPPRESSION

Limits the number of times a specific warning or error message is reported.

Syntax

```
MESSAGE_SUPPRESSION: ID_1:limit1 ID_2:limit2 ...
```

Arguments

Argument	Description
ID_1, ID_2	Affected message ID
limit1, limit2	Maximum number of times to report the specified message

Description

In cases where the StarRC tool generates a large number of warning, error, or information messages, you might want to limit the number of times that similar messages (messages that have the same ID) are issued.

The `MESSAGE_SUPPRESSION` command limits the number of times that specific messages are issued. The limit applies only to messages whose IDs are listed in the command. If the same message ID appears in more than one `MESSAGE_SUPPRESSION` command, the limit specified in the last statement takes precedence.

The similar `MESSAGE_SUPPRESSION_LIMIT` command applies a limit to all warning, error, and information messages and does not require you to name specific message IDs.

Limits for the following messages are fixed at 1000 and are not affected by the `MESSAGE_SUPPRESSION` or `MESSAGE_SUPPRESSION_LIMIT` commands:

- Shorts message EX-503
- Fill shorts message EX-505
- SMIN violation messages EX-792 and EX-356
- Via violation message EX-714

This feature applies only to SX, EX, and GRD messages, which are messages issued by the StarRC tool. During an extraction run, messages with different prefixes might appear.

Errors

By default, the StarRC tool issues up to 100 messages of the same type. You can increase or decrease this limit. However, if you set the limit to a value greater than 1000, runtime

might be increased and the tool issues an SX-3253 warning message. Values greater than 100,000 are set to 100,000 and the tool issues an SX-3252 warning message.

Examples

The following command causes stops reporting SX-2549 messages after the fifth occurrence and EX-269 messages after the twentieth occurrence:

```
MESSAGE_SUPPRESSION: SX-2549:5 EX-269:20
```

See Also

- MESSAGE_SUPPRESSION_LIMIT

Feedback

MESSAGE_SUPPRESSION_LIMIT

Limits the number of times any single warning, error, or information message is reported.

Syntax

```
MESSAGE_SUPPRESSION_LIMIT: count_value
```

Arguments

Argument	Description
count_value	Maximum number of times to report any single message Default: 100

Description

If the StarRC tool generates a large number of warning, error, or information messages, you might want to limit the number of similar messages. Use the `MESSAGE_SUPPRESSION_LIMIT` command to limit the number of times that messages with the same ID are reported. The limit applies to all SX, EX, and GRD messages, with the following exceptions:

- Messages specified in a `MESSAGE_SUPPRESSION` command
- The following messages, whose limits are fixed at 1000:
  - Shorts message EX-503
  - Fill shorts message EX-505
  - SMIN violation messages EX-792 and EX-356
  - Via violation message EX-714

If you specify a value greater than 1000, the tool issues an SX-3253 warning. The tool sets values greater than 100,000 to 100,000 and issues an SX-3252 warning.

Examples

The following commands cause the tool to stop reporting most messages after the tenth occurrence, but to report EX-269 messages up to twenty times:

```
MESSAGE_SUPPRESSION_LIMIT: 10
MESSAGE_SUPPRESSION: EX-269:20
```

See Also

- MESSAGE_SUPPRESSION

Feedback

METAL_FILL_BLOCK_MAPPING_FILE

Specifies a file that maps metal fill macros to design macros for LEF/DEF gate-level flows.

Syntax

```
METAL_FILL_BLOCK_MAPPING_FILE: fill_map_file
```

Arguments

Argument	Description
fill_map_file	Metal fill mapping file name Default: none

Description

The METAL_FILL_BLOCK_MAPPING_FILE command specifies the name of a mapping file that associates design macros names with fill macro names. This command is valid only for LEF/DEF designs.

The fill mapping file uses the following syntax, where the two names are separated by one or more spaces:

```
* Metal fill macro assignments
design_macro_name     fill_macro_name
```

The first entry on a line specifies the design macro to which the fill is to be applied. The second entry is the associated metal fill macro. If a design name is repeated, the first definition is honored.

Denote a comment line in the file with an initial asterisk (*).

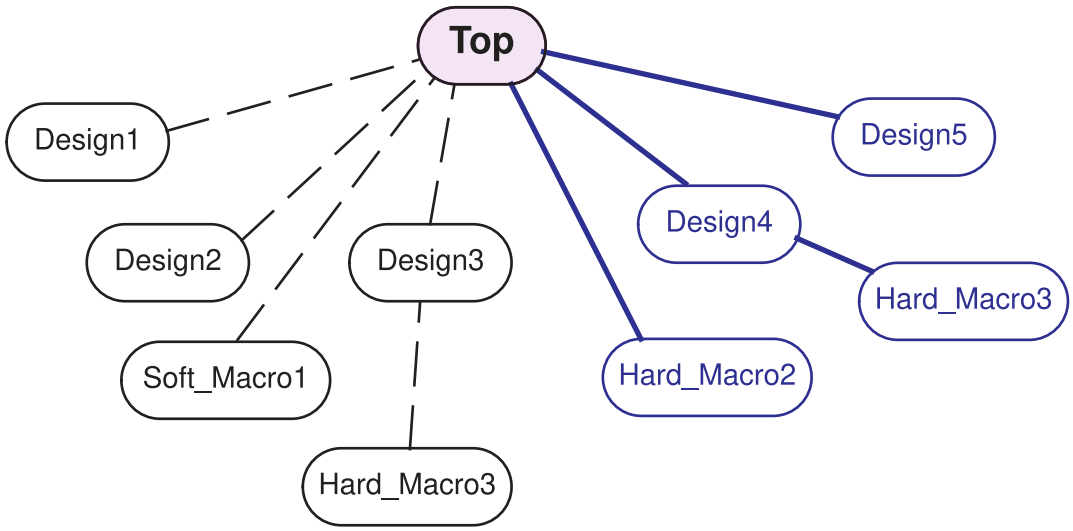
Examples

For example, a metal fill mapping file for the design shown in Figure 205 might be as follows:

```
Hard_Macro2 Hard_Macro2_fill
Hard_Macro3 Hard_Macro3_fill
Design5 Design5_fill
Design4 Design4_fill
```

Feedback

Figure 205 Example Hierarchical Design



In this example, design macros Hard_Macro2, Hard_Macro3, Design4, and Design5 are targeted for hierarchical fill. The metal fill mapping file names the fill macros to be used with each of these design macros.

The StarRC tool handles the metal fill for the remaining design macros (such as Design1 and Soft_Macro1) using the default method, which follows the hierarchy information provided in the metal fill top cell.

The following usage notes apply:

- A fill macro named in the metal fill mapping file must not be referenced in its parent design's fill cell.

  In this example, fill cell Design4_fill must not reference fill cell Hard_Macro3_fill, because the mapping file controls the associations. The StarRC tool issues a warning message if this improper reference is detected.

  However, a fill cell for Design3 can contain a reference for fill cell Hard_Macro3_fill, because the fill for Design3 and its child cells is not controlled by the mapping file.

- The top-level fill macro should not contain references to child design fill macros that are named in the mapping file.

Feedback

See Also

- METAL_FILL_GDS_FILE
- METAL_FILL_OASIS_FILE
- Real Metal Fill

Feedback

METAL_FILL_BLOCK_NAME

Changes the top-level block name in the metal fill database to match the corresponding block name in the design database.

Syntax

```
METAL_FILL_BLOCK_NAME: top_block_name
```

Arguments

Argument	Description
top_block_name	The top-level block name in the metal fill database

Description

By default, the StarRC tool assumes the top-level block name in a metal fill database is the same as the top-level block name in the design database. If the names are different, use the METAL_FILL_BLOCK_NAME command to specify the top-level block name in the metal fill database.

If the provided block name does not match the block name in the design database, the run terminates and the tool issues an error message.

The METAL_FILL_BLOCK_NAME command is valid for both OASIS and GDSII files. The METAL_FILL_BLOCK_NAME command takes precedence over the METAL_FILL_GDS_BLOCK command if both are present in the command file.

Examples

The following example specifies the top-level block name in the fill design:

```
METAL_FILL_BLOCK_NAME: top_fill_block1
```

See Also

- METAL_FILL_BLOCK_MAPPING_FILE
- METAL_FILL_GDS_BLOCK
- METAL_FILL_GDS_FILE
- METAL_FILL_GDS_FILE_NET_NAME
- Real Metal Fill

Feedback

METAL_FILL_GDS_BLOCK

Changes the top-level block name in the metal-fill GDSII database to match the corresponding block name in the main design database.

Syntax

```
METAL_FILL_GDS_BLOCK: top_block_name
```

Arguments

Argument	Description
top_block_name	The top-block name in the main design database

Description

By default, the StarRC tool assumes the top-level block name in a GDSII metal fill database is the same as the top-level block name in the design database. If the names are different, use the METAL_FILL_GDS_BLOCK command or the METAL_FILL_BLOCK_NAME command to change the top-level block name in the GDSII metal fill database. If both commands are present in the command file, the METAL_FILL_BLOCK_NAME command takes precedence.

If the provided block name does not match the block name in the design database, the run terminates and the tool issues an error message.

To change the block name for an OASIS format metal fill database, use the METAL_FILL_BLOCK_NAME command.

Examples

The following example specifies the top-level block name in the GDSII database name in the design:

```
METAL_FILL_GDS_BLOCK: top_block1
```

See Also

- METAL_FILL_BLOCK_NAME
- METAL_FILL_GDS_FILE
- METAL_FILL_GDS_FILE_NET_NAME
- Real Metal Fill

Feedback

METAL_FILL_GDS_FILE

Specifies one or more GDSII files that contains metal fill data.

Syntax

```
METAL_FILL_GDS_FILE: file1 file2 ...
```

Arguments

Argument	Description
file1 file2 ...	GDSII files that contain metal fill data Default: none

Description

The METAL_FILL_GDS_FILE command lists GDSII files that contain metal fill data. This command supports both hierarchical and flat GDSII files, which can also be compressed. The METAL_FILL_GDS_FILE command can appear multiple times in a command file.

You can provide metal fill information from any combination of the following sources:

- The design database (Fusion Compiler, IC Compiler II, Milkyway, or LEF/DEF designs)
- One or more GDSII files, by using the METAL_FILL_GDS_FILE command

  In this case, you must also provide layer mapping information by using the GDS_LAYER_MAP_FILE command.
- One or more OASIS files, by using the METAL_FILL_OASIS_FILE command

  In this case, you must also provide layer mapping information by using the OASIS_LAYER_MAP_FILE command.

Shapes in a GDSII or OASIS file are treated as metal fill objects for extraction if the following conditions are met. All other data in the file is ignored.

- The shapes are on a layer that is listed in the respective mapping file.
- The shapes are referenced by the top-level block definition in the metal fill file or by a child cell of the top-level block.
- The top-level block name of the metal fill data matches the top-level block name of the design database. You can change the top-level block name for the metal fill data by using the METAL_FILL_BLOCK_NAME or METAL_FILL_GDS_BLOCK commands.

The METAL_FILL_POLYGON_HANDLING command applies to all metal fill shapes.

Feedback

Note:

The StarRC tool does not support a flow in which metal fill polygons provided in a GDSII file connect to more than one power net.

Metal Fill Statistics Reports

Reports generated by setting the REPORT_METAL_FILL_STATISTICS command to YES contain information for metal fill from all sources. Standard reporting is available for fill read from a design database. Advanced reporting is available for fill read from GDSII or OASIS files.

Handling Duplicate Cells

Within a design database, a fill cell is local to the design that contains it. The tool that creates the designs can handle duplicate fill cells by importing the fill cell and its libraries separately for each design that references it.

However, when metal fill is obtained from GDSII or OASIS files, the StarRC tool imports only the first instance of each unique fill cell found in those files. If duplicate cell definitions exist, the tool chooses the first cell definition encountered. Therefore the order of specifying commands and file names in the StarRC command file might affect the result, as follows:

- If the command file contains instances of both the METAL_FILL_GDS_FILE and METAL_FILL_OASIS_FILE commands, the tool executes the METAL_FILL_GDS_FILE commands first.
- If a METAL_FILL_GDS_FILE command or METAL_FILL_OASIS_FILE command contains multiple files in its argument list, the tool reads the files in the order specified.
- If the command file contains multiple instances of the METAL_FILL_GDS_FILE command, the tool reads the files in the order specified. In other words, the tool reads all files in the first instance of the command, followed by all files in the second instance, and so on.
- The tool reads OASIS files from multiple instances of the METAL_FILL_OASIS_FILE command in a similar way, but only after all GDSII files are read.

See Also

- GDS_FILE
- GDS_LAYER_MAP_FILE
- METAL_FILL_GDS_FILE_NET_NAME
- METAL_FILL_POLYGON_HANDLING
- REPORT_METAL_FILL_STATISTICS

Feedback

- METAL_FILL_OASIS_FILE
- The StarXtract -gdscheck Option
- Real Metal Fill

Feedback

METAL_FILL_GDS_FILE_NET_NAME

Ties metal fill polygons to a power or ground net.

Syntax

```
METAL_FILL_GDS_FILE_NET_NAME: net_name
```

Arguments

Argument	Description
net_name	The layout net name Default: none

Description

You can use the METAL_FILL_GDS_FILE_NET_NAME command to tie metal fill polygons to a power or ground net.

The METAL_FILL_GDS_FILE_NET_NAME command

- Must be specified together with METAL_FILL_GDS_FILE and METAL_FILL_POLYGON_HANDLING: GROUNDED
- Cannot be used with the METAL_FILL_OASIS_FILE_NET_NAME command

See Also

- GDS_LAYER_MAP_FILE
- METAL_FILL_GDS_FILE
- METAL_FILL_POLYGON_HANDLING
- Real Metal Fill

Feedback

METAL_FILL_GDS_MAG

Specifies the scaling factor that is applied to the GDSII metal fill data.

Syntax

```
METAL_FILL_GDS_MAG: factor
```

Arguments

Argument	Description
factor	Magnification factor Default: 1.0

Description

The METAL_FILL_GDS_MAG command specifies the scaling factor that is applied to the GDSII metal fill data.

The metal fill offset specified by the METAL_FILL_GDS_OFFSET command is not multiplied by the scaling factor.

Note:

The METAL_FILL_GDS_MAG command cannot be used with the METAL_FILL_OASIS_MAG command.

Examples

In the following example, the GDSII metal fill data length and width are multiplied by a factor of 0.8:

```
METAL_FILL_GDS_MAG: 0.8
```

The total entire area of the design would therefore be scaled by a factor of 0.64.

See Also

- METAL_FILL_GDS_FILE
- METAL_FILL_GDS_OFFSET
- Real Metal Fill

Feedback

METAL_FILL_GDS_OFFSET

Specifies the origin of the metal fill GDSII file.

Syntax

```
METAL_FILL_GDS_OFFSET: x_coordinate y_coordinate
```

Arguments

Argument	Description
x_coordinate	X-coordinate Units: microns Default: 0.0
y_coordinate	Y-coordinate Units: microns Default: 0.0

Description

The METAL_FILL_GDS_OFFSET command specifies the coordinates of the origin of the metal fill GDSII file. This command does not affect the magnification factor behavior.

Note:

The METAL_FILL_GDS_OFFSET command cannot be used with the METAL_FILL_OASIS_OFFSET command.

Examples

```
METAL_FILL_GDS_OFFSET: 10.8 5.3
```

See Also

- METAL_FILL_GDS_FILE
- METAL_FILL_GDS_FILE_NET_NAME
- METAL_FILL_POLYGON_HANDLING
- Real Metal Fill

Feedback

METAL_FILL_OASIS_FILE

Specifies one or more OASIS files that contain metal fill data.

Syntax

```
METAL_FILL_OASIS_FILE: file1 file2 ...
```

Arguments

Argument	Description
file1 file2 ...	OASIS files that contain metal fill data Default: none

Description

The METAL_FILL_OASIS_FILE command lists OASIS files that contain metal fill data. This command supports both hierarchical and flat OASIS files. The METAL_FILL_OASIS_FILE command can appear multiple times in a command file.

You can obtain metal fill information from any combination of the following sources:

- The design database (Fusion Compiler, IC Compiler II, Milkyway, or LEF/DEF designs)
- One or more GDSII files, by using the METAL_FILL_GDS_FILE command

  In this case, you must also provide layer mapping information by using the GDS_LAYER_MAP_FILE command.
- One or more OASIS files, by using the METAL_FILL_OASIS_FILE command

  In this case, you must also provide layer mapping information by using the OASIS_LAYER_MAP_FILE command.

Shapes in a GDSII or OASIS file are treated as metal fill objects for extraction if the following conditions are met. All other data in the file is ignored.

- The shapes are on a layer that is listed in the respective mapping file.
- The shapes are referenced by the top-level block definition in the metal fill file or by a child cell of the top-level block.
- The top-level block name of the metal fill data matches the top-level block name of the design database. You can change the top-level block name for the metal fill data by using the METAL_FILL_BLOCK_NAME or METAL_FILL_GDS_BLOCK commands.

The METAL_FILL_POLYGON_HANDLING command applies to all metal fill shapes.

Feedback

Note:

The StarRC tool does not support a flow in which metal fill polygons provided in a GDSII file connect to more than one power net.

Metal Fill Statistics Reports

Set the REPORT_METAL_FILL_STATISTICS command to YES to report information about metal fill from all sources. Standard reporting is available for fill read from a design database. Advanced reporting is available for fill read from GDSII or OASIS files.

Handling Duplicate Cells

Within a design database, a fill cell is local to the design that contains it. The tool that creates the designs can handle duplicate fill cells by importing the fill cell and its libraries separately for each design that references it.

However, when metal fill is obtained from GDSII or OASIS files, the StarRC tool imports only the first instance of each unique fill cell found in those files. If duplicate cell definitions exist, the tool chooses the first cell definition encountered. Therefore the order of specifying commands and file names in the StarRC command file might affect the result, as follows:

- If the command file contains instances of both the METAL_FILL_GDS_FILE and METAL_FILL_OASIS_FILE commands, the tool executes the METAL_FILL_GDS_FILE commands first.
- If a METAL_FILL_GDS_FILE command or METAL_FILL_OASIS_FILE command contains multiple files in its argument list, the tool reads the files in the order specified.
- If the command file contains multiple instances of the METAL_FILL_GDS_FILE command, the tool reads the files in the order specified. In other words, the tool reads all files in the first instance of the command, followed by all files in the second instance, and so on.
- The tool reads OASIS files from multiple instances of the METAL_FILL_OASIS_FILE command in a similar way, but only after all GDSII files are read.

See Also

- OASIS_LAYER_MAP_FILE
- OASIS_FILE
- METAL_FILL_BLOCK_NAME
- METAL_FILL_GDS_FILE
- METAL_FILL_GDS_BLOCK
- METAL_FILL_POLYGON_HANDLING

- REPORT_METAL_FILL_STATISTICS
- Real Metal Fill

Feedback

METAL_FILL_OASIS_FILE_NET_NAME

Ties metal fill polygons to a power or ground net.

Syntax

```
METAL_FILL_OASIS_FILE_NET_NAME: net_name
```

Arguments

Argument	Description
net_name	The layout net name Default: none

Description

You can use the METAL_FILL_OASIS_FILE_NET_NAME command to tie metal fill polygons to a power or ground net.

The METAL_FILL_OASIS_FILE_NET_NAME command

- Must be specified together with METAL_FILL_OASIS_FILE and METAL_FILL_POLYGON_HANDLING: GROUNDED
- Cannot be used with the METAL_FILL_GDS_FILE_NET_NAME command

See Also

- METAL_FILL_OASIS_FILE
- METAL_FILL_POLYGON_HANDLING
- OASIS_LAYER_MAP_FILE
- Real Metal Fill

Feedback

METAL_FILL_OASIS_MAG

Specifies the scaling factor that is applied to the OASIS metal fill data.

Syntax

```
METAL_FILL_OASIS_MAG: factor
```

Arguments

Argument	Description
factor	Magnification factor Default: 1.0

Description

The METAL_FILL_OASIS_MAG command specifies the scaling factor that is applied to the OASIS metal fill data.

The metal fill offset specified by the METAL_FILL_GDS_OFFSET command is not multiplied by the scaling factor.

Note:

The METAL_FILL_OASIS_MAG command cannot be used with the METAL_FILL_GDS_MAG command.

Examples

In the following example, the OASIS metal fill data length and width are multiplied by a factor of 0.8:

```
METAL_FILL_OASIS_MAG: 0.8
```

The total entire area of the design would therefore be scaled by a factor of 0.64.

See Also

- METAL_FILL_OASIS_FILE
- METAL_FILL_OASIS_OFFSET
- Real Metal Fill

Feedback

METAL_FILL_OASIS_OFFSET

Specifies the origin of the metal fill OASIS file.

Syntax

```
METAL_FILL_OASIS_OFFSET: x_coordinate y_coordinate
```

Arguments

Argument	Description
x_coordinate	X-coordinate Units: microns Default: 0.0
y_coordinate	Y-coordinate Units: microns Default: 0.0

Description

The METAL_FILL_OASIS_OFFSET command specifies the coordinates of the origin of the metal fill OASIS file. This command does not affect the magnification factor behavior.

The METAL_FILL_OASIS_OFFSET command cannot be used with the METAL_FILL_GDS_OFFSET command.

Examples

```
METAL_FILL_OASIS_OFFSET: 10.8 5.3
```

See Also

- METAL_FILL_OASIS_FILE
- METAL_FILL_OASIS_FILE_NET_NAME
- METAL_FILL_POLYGON_HANDLING
- Real Metal Fill

Feedback

METAL_FILL_POLYGON_HANDLING

Specifies the treatment of metal fill polygons.

Syntax

```
METAL_FILL_POLYGON_HANDLING: IGNORE | GROUNDED | FLOATING | AUTOMATIC
```

Arguments

Argument	Description
IGNORE (the default)	Performs resistance and capacitance extraction without considering the effect of metal fill polygons. In a Milkyway design, metal fill polygons are ignored only if they have the correct ROUTE_TYPE specification.
GROUNDED	Performs extraction by treating metal fill as grounded. By default, metal fill polygons are treated as connected to ground during extraction, unless the Milkyway or LEF/DEF design database ties the metal fill polygons to a specific power or ground net or the METAL_FILL_GDS_FILE_NET_NAME command is specified.
FLOATING	Performs extraction by treating the metal fill as floating. This argument overrides the type of metal fill polygons provided in the input database. This means that even though the input database has grounded fill polygons, the METAL_FILL_POLYGON_HANDLING command extracts the capacitance as if the fill polygons were floating and ignores the existence of fill objects in the database.
AUTOMATIC	Performs automatic extraction by treating metal fill as floating or grounded. For LEF/DEF designs, this argument parses the DEF file and translates fill polygons based on the section in which they appear in the DEF file. For Milkyway designs, this argument translates fills based on the ROUTE_TYPE value attached to a net ID or the absence of a net ID (floating). Both floating and grounded fills are allowed in the same design.

Description

The METAL_FILL_POLYGON_HANDLING command specifies how to treat metal fill polygons during extraction. The metal fill must come from the design or from metal fill input files in GDSII or OASIS format.

Note:

The METAL_FILL_POLYGON_HANDLING command does not apply to metal fill derived from floating nets. In that case, use the TRANSLATE_FLOATING_AS_FILL command instead.

If metal fill polygons are written into the FILL view in the Milkyway database, you must also set the MILKYWAY_ADDITIONAL_VIEWS command.

Feedback

Table 71 lists StarRC commands related to the METAL_FILL_POLYGON_HANDLING command for designs that provide metal fill in GDSII files. Similar commands are available for OASIS files.

Table 71 Commands Related to the METAL_FILL_POLYGON_HANDLING Command

Related command	Description
METAL_FILL_GDS_FILE_NET_NAME	The METAL_FILL_GDS_FILE_NET_NAME command is required when you want to tie metal fill polygons to a power or ground net. The net name should match the layout net name. This command works only when the METAL_FILL_POLYGON_HANDLING command is set to GROUNDED.
METAL_FILL_GDS_FILE	The METAL_FILL_GDS_FILE command supports either hierarchical or flat GDSII files. The cell name of the GDS file must be the same as the BLOCK name or top cell name of the design.
GDS_LAYER_MAP_FILE	The GDS_LAYER_MAP_FILE command specifies the file that contains information about the mapping between the GDSII layer number and the layer name in the design database.
GDS_FILE	If the GDS_FILE command is specified for a LEF/DEF database, a single unified layer mapping file should be used for both GDSII files.

Examples

This command treats the metal fill polygons as grounded:

```
METAL_FILL_POLYGON_HANDLING: GROUNDED
```

See Also

- GDS_FILE
- GDS_LAYER_MAP_FILE
- METAL_FILL_GDS_FILE
- METAL_FILL_GDS_FILE_NET_NAME
- MILKYWAY_ADDITIONAL_VIEWS
- TRANSLATE_FLOATING_AS_FILL
- Real Metal Fill

Feedback

METAL_SHEET_OVER_AREA

Specifies an area for sheet metal coupling capacitance measurement.

Syntax

```
METAL_SHEET_OVER_AREA: layer_name X1 Y1 X2 Y2
```

Arguments

Argument	Description
layer_name	Metal layer name Default: none
X1 Y1	Lower-left coordinates of the area Units: microns
X2 Y2	Upper-right coordinates of the area Units: microns

Description

Use this command to associate a sheet of metal to a user-defined net name and output suffix. You can use the command multiple times to specify multiple metal sheets. You can optionally specify the `SHEET_COUPLE_TO_NET_LEVEL` command to enable a net name suffix.

You must verify that the sheet metal areas do not cause metal shorts. The StarRC tool does not check for areas of metal that overlay each other. The tool checks that the specified layer is a metal layer and that the bounding box coordinates are correct.

Examples

```
METAL_SHEET_OVER_AREA: METAL2 0 0 100 100
METAL_SHEET_OVER_AREA: METAL2 200 200 400 400
METAL_SHEET_OVER_AREA: METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL: YES
```

See Also

- SHEET_COUPLE_TO_NET
- SHEET_COUPLE_TO_NET_LEVEL

Feedback

MILKYWAY_ADDITIONAL_VIEWS

Specifies a Milkyway view.

Syntax

```
MILKYWAY_ADDITIONAL_VIEWS: view_name
```

Arguments

Argument	Description
view_name	Name of the additional view Default: none

Description

Milkyway stores design data in different files called *views* inside a generated Milkyway library. Use this command to read views other than `CEL`, `FRAM`, `TIM`, or `PWR` views. The previously listed views are automatically read. The `MILKYWAY_ADDITIONAL_VIEWS` command reads an additional view.

See the Milkyway documentation for a complete list of output views.

Examples

```
MILKYWAY_ADDITIONAL_VIEWS: FILL
```

See Also

- METAL_FILL_POLYGON_HANDLING

Feedback

MILKYWAY_CELL_VIEW

Specifies a list of cells for which to use the layout cell view.

Syntax

```
MILKYWAY_CELL_VIEW: list_of_cells
```

Arguments

Argument	Description
list_of_cells	List of cells for which to use the layout cell view during extraction, if available Default: none

Description

This command specifies a white-space-delimited list of cells for which the StarRC tool uses the layout cell view (Milkyway CEL view) during extraction, if that view is available. You can specify this command multiple times in a single command file. The asterisk (`*`) and question mark (`?`) wildcard characters are accepted.

Note:

This command applies to skip cells and their child cells only; the CEL view is always used for non `skipped cell` masters.

For skip cells not on this list, the Milkyway FRAM view represents the cell contents during extraction. The FRAM view typically contains all pin shapes and obstructions.

The StarRC tool attempts to translate the Milkyway cell view for each skip cell on this list. The cell view contains the actual physical layout, including nonroute layers. Specifying a cell in this list automatically includes all child cells. If a cell view cannot be found for a cell contained in this list, the tool issues a warning and reverts to the FRAM view for that cell and all child cells.

Examples

```
MILKYWAY_CELL_VIEW: cell1 cell2 cell3
MILKYWAY_CELL_VIEW: cellA cellB cell? *C
MILKYWAY_CELL_VIEW: *
```

See Also

- SKIP_CELLS

Feedback

MILKYWAY_DATABASE

Specifies the location of the input Milkyway layout database.

Syntax

```
MILKYWAY_DATABASE: layout_library
```

Arguments

Argument	Description
layout_library	The name of the layout library from the Milkyway database Default: none

Description

The `MILKYWAY_DATABASE` command is mandatory for a Milkyway extraction flow.

You must specify the block for extraction with the `BLOCK` command.

See Also

- BLOCK

Feedback

MILKYWAY_EXPAND_HIERARCHICAL_CELLS

Flattens any routed cell instance that has the cell type or property *macro*.

Syntax

```
MILKYWAY_EXPAND_HIERARCHICAL_CELLS: YES | NO
```

Arguments

Argument	Description
YES	Flattens the cell types that are routed in the Milkyway database and to run extraction
NO (default)	Does not flatten macro or routed cells

Description

For this command, “routed” means any cell that contains one or more nets or more than one instance placement. Any other cell type remains a member of the list of skip cells specified by the SKIP_CELLS command.

The MILKYWAY_EXPAND_HIERARCHICAL_CELLS command automatically configures the skip cells list and takes precedence over the SKIP_CELLS command.

See Also

- SKIP_CELLS

Feedback

MILKYWAY_EXTRACT_VIEW

Selects the XTR (Milkyway extract view) layout description as the input for extraction.

Syntax

```
MILKYWAY_EXTRACT_VIEW: YES | NO
```

Arguments

Argument	Description
YES	StarRC reads the Milkyway XTR view
NO (default)	StarRC does not read the Milkyway XTR view

Description

This command selects the XTR (Milkyway extract view) layout description as the input for extraction. This command is mandatory for a Hercules flow.

Errors

To read Hercules output, you must specify the `MILKYWAY_EXTRACT_VIEW: YES` command. If the `MILKYWAY_EXTRACT_VIEW` command is not set, the StarRC tool treats the Milkyway library that was generated by Hercules as if it were a library generated by Synopsys physical design tools and displays the following message:

```
WARNING: cannot open milkyway cell top:CEL!
```

See Also

- BLOCK
- MILKYWAY_DATABASE
- POWER_NETS

Feedback

MILKYWAY_REF_LIB_MODE

Specifies the order in which libraries are searched for a cell master.

Syntax

```
MILKYWAY_REF_LIB_MODE: NONE | HIER | FILE
```

Arguments

Argument	Description
`NONE` (default)	The cell is first searched for in the reference library. The search order for the reference library is the same as it is in the main design library.
`HIER`	The child cell is searched for in the same library as its parent library. If the child cell is not found, the default mode is used.
`FILE`	A reference control file in the main library specifies which reference library is checked first. The specified order is followed to find and open the cell.

Description

When extracting Milkyway databases, the `MILKYWAY_REF_LIB_MODE` command controls the search preference among libraries and reference libraries for the cell master.

Note:

If you use the Fusion Compiler or IC Compiler tool for place and route, you should specify the `MILKYWAY_REF_LIB_MODE: HIER` command to follow the same search sequence as the Fusion Compiler or IC Compiler tool.

In the HIER mode, the CEL views are uniquified by appending special characters to the cell names so that they are hierarchically separated from each other in case of name collisions across different libraries. This uniquification might affect the names used in skip cells when exploding a specific cell. For example, if a cell named macroA needs to be exploded or flattened, you must use the `SKIP_CELLS: macroA*` command to explode the cell across different libraries.

Other tools use different commands or options to specify the library search order. See the documentation for those tools for more details to ensure that you specify a consistent search order throughout your entire design flow.

Feedback

Examples

In the library structure shown in Example 27, if you specify `MILKYWAY_REF_LIB_MODE: NONE`, the StarRC tool uses Cell A in ref lib 1. If you specify `MILKYWAY_REF_LIB_MODE: HIER`, the tool uses Cell A in the main library.

Example 27

```
[ Top/top lib ]  --[A1/(instantiated from cell A)]
         |                        - Cell A
         |----------[ lib1/ref lib]  - cell A
```

In the library structure shown in Example 28, the tool uses ref lib/lib1 cell A for instance A1.

Example 28

```
[ Top/top lib ]  --[A1/(instantiated from cell A)]
         |----------[ lib1/ref lib]  - cell A
         |----------[ lib2/ref lib]  - cell A
```

In the library structure shown in Example 29, the tool uses ref lib/lib 1.

Example 29

```
[ Top/top lib ]  --[A1/(instantiated from cell A]
         |----------[ lib1/ref lib]  - cell A
                              |----------[ lib2/ref lib]  - cell A
```

See Also

- MILKYWAY_DATABASE

Feedback

MILKYWAY_SHOW_CELL_INFO_DETAIL

Writes information about every translated cell into a log file.

Syntax

```
MILKYWAY_SHOW_CELL_INFO_DETAIL: YES | NO
```

Arguments

Argument	Description
YES	Writes information about every translated cell into a log file
NO (default)	Does not report cell details

Description

Milkyway designs contain data in different files known as views, such as the CEL and FRAM views. One cell might have many different views. In addition, different design libraries might use the same cell name. Many commands in the StarRC command file affect which libraries, cells, and views are translated. Therefore, it might be difficult to know exactly which design files are used in an extraction run.

To obtain detailed information about every translated cell, set the MILKYWAY_SHOW_CELL_INFO_DETAIL command to YES.

If the input data contains GDSII or OASIS data, the StarRC tool writes the information into the summary/cells.sum file. Otherwise, the tool writes the information into the star/cell_info.detail file.

Examples

The following lines are examples of the information that might be included in the cell_info.detail file. Each line contains the cell name, the view name (highlighted in color in this example), and the library name.

```
Cell Info: FILL2BWP16P90ULVT FRAM /slowfs/dept5242t/MW/tcbn16ff11bwp 16p
Cell Info: mimcap_unitcell FRAM /slowfs/dept52425t/MW_w_power
Cell Info: toprt CEL /na4apd/starrc/mw/xtdesign
Cell Info: toprt FILL /na4apd/starrc/mw/xtdesign
```

See Also

- MILKYWAY_DATABASE

Feedback

MILKYWAY_USE_CELL_PINS

Maintains consistency in pin names from the block-level and top-level extraction by using pins from the CEL view.

Syntax

```
MILKYWAY_USE_CELL_PINS: | NO | YES
```

Arguments

Argument	Description
`NO` (default)	Uses pins from the FRAM view.
`YES`	Uses pins preferably from the CEL view. If there are no pins in the CEL view, uses pins from the FRAME view.

Description

In a Milkyway database, pin names in the FRAM and CEL views might not be consistent. To avoid inconsistencies between pin names from block-level and top-level extractions in the multiple physical-pins hierarchical flow, set the `MILKYWAY_USE_CELL_PINS` command to `YES` to use pin names from CEL views preferentially. If the CEL view does not contain a pin, then pin names are taken from the FRAM view.

For more information, see Multiple Physical-Pins Hierarchical Flow.

See Also

- MULTI_PHYSICAL_PINS_PREFIX

Feedback

MODEL_TYPE

Specifies whether the reference model comes from the layout or the schematic.

Syntax

```
MODEL_TYPE: LAYOUT | SCHEMATIC
```

Arguments

Argument	Description
`LAYOUT` (default)	Specifies that the reference model has been generated from a layout
`SCHEMATIC`	Specifies that the reference model has been generated from a schematic. This setting is not allowed with the `XREF:NO` command.

Description

This command specifies whether the reference model in the `HN_NETLIST_MODEL_NAME`, `RETAIN_CAPACITANCE_CAP_MODELS`, or `HN_NETLIST_SPICE_TYPE` commands comes from the layout or the schematic

The StarRC tool reports the layout net names generated during ideal layout extraction, as shown in Table 72.

Table 72 Effect of the XREF Command

XREF command setting	Behavior
`XREF: YES | COMPLETE`	Prints schematic model name in the parasitic netlist
`XREF: NO`	Prints the layout model name (default)
`XREF: YES`	Sets `XREF_USE_LAYOUT_DEVICE_NAME: YES` in the command file for layout model name output

Usage for Calibre Connectivity Interface Extraction

The Calibre tool supports multiple devices that each have their own seed (device) layers. Devices might share the same model name for layout versus schematic (LVS) analysis but have different model names for use by downstream tools.

The StarRC tool treats the device model name as the layout model name and the netlist model name as the schematic model name.

If the MODEL_TYPE command is not set in the StarRC command file for a Calibre Connectivity Interface flow, the StarRC tool sets the MODEL_TYPE command to SCHEMATIC and looks for the netlist model name. If you set the MODEL_TYPE command to LAYOUT, the StarRC tool looks for the device model name.

Examples

```
MODEL_TYPE: SCHEMATIC
HN_NETLIST_MODEL_NAME: myrcxtmodel mysim_modelname
```

See Also

- XREF
- HN_NETLIST_MODEL_NAME
- HN_NETLIST_SPICE_TYPE
- RETAIN_CAPACITANCE_CAP_MODELS

Feedback

MOS_GATE_CAPACITANCE

Specifies a global loading capacitance per unit area.

Syntax

```
MOS_GATE_CAPACITANCE: load_cap
```

Arguments

Argument	Description
load_cap	MOS gate capacitance Units: farads per square micron Default: 1e-15, but defaults to zero for advanced device property extraction

Description

Specifies a global loading capacitance per unit area (in square microns) for MOS gate terminals in the Detailed Standard Parasitic Format (DSPF) and SPEF connectivity sections (*|I and *I, respectively) of the output parasitic netlist. Only devices generated by the Hercules commands NMOS and PMOS are assigned this capacitance. In addition, all MOS gates are written in the netlist with direction “I”.

Feedback

MOS_GATE_CAP_DISTRIBUTION

Specifies whether to place gate capacitance at the center or at the ends of the gate polygon. Valid only for transistor-level flows.

Syntax

```
MOS_GATE_CAP_DISTRIBUTION: ENDS | CENTER
```

Arguments

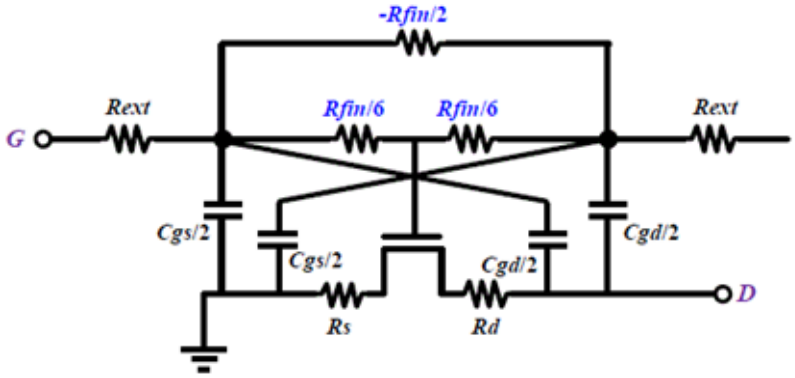
Argument	Description
ENDS (default)	Gate capacitance is distributed between the ends of the gate polygon
CENTER	Gate capacitance is placed at the center of the gate polygon

Description

The MOS_GATE_CAP_DISTRIBUTION command specifies where the gate-to-drain and gate-to-source capacitances are attached to the gate polygon.

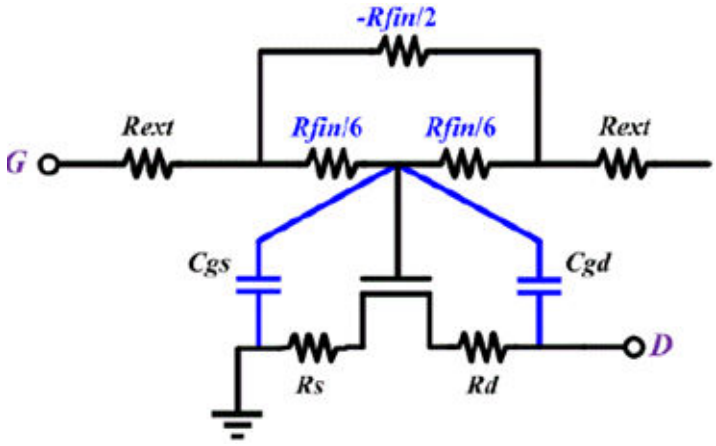
Figure 206 illustrates the default behavior, in which the gate capacitances are split and distributed between the two ends of the gate polygon. Figure 207 shows the effect of setting the MOS_GATE_CAP_DISTRIBUTION command to CENTER. In this case, all capacitances from the gate polygon to other layers are attached to the center of the gate polygon.

Figure 206 Command Setting ENDS: Capacitance Distributed at Polygon Ends



Feedback

Figure 207 Command Setting CENTER: Capacitance Placed at Polygon Center



In these examples, the resistors with values of Rfin/6 and -Rfin/2 are created when the `MOS_GATE_DELTA_RESISTANCE` command is used. Without this command, the resistor with a value of -Rfin/2 does not exist and the resistors with a value of Rfin/6 instead have a value of Rfin/2.

See Also

- MOS_GATE_DELTA_RESISTANCE

Feedback

MOS_GATE_DELTA_RESISTANCE

Specifies whether to extract the gate resistance of MOS devices as a delta network.

Syntax

```
MOS_GATE_DELTA_RESISTANCE: YES | NO
```

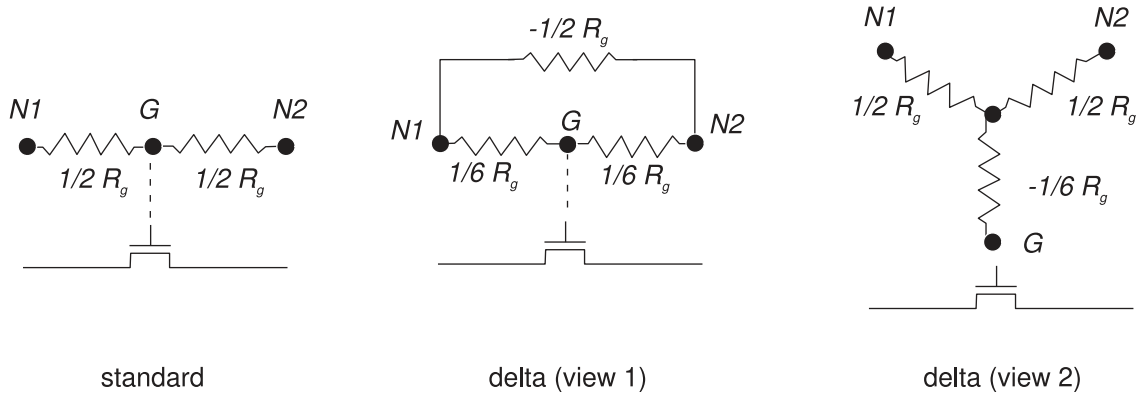
Arguments

Argument	Description
YES	Uses the delta network model for MOS gate resistance. Does not apply to X devices.
NO (default)	Uses the standard gate resistance model

Description

This command changes the gate resistance model for MOS devices to a delta network. Figure 208 shows the standard model and two equivalent views of the delta model. Nodes N1 and N2 represent the ends of the gate polygon, while node G represents the ideal gate terminal location.

Figure 208 Standard Gate Resistance and Two Views of Delta Resistance Network



standard delta (view 1) delta (view 2)

In the standard model, the resistance from node N1 to node N2 is Rg and the resistance from either node to node G is 1/2 Rg. The value of 1/2 Rg appears in the parasitic netlist.

If the MOS_GATE_DELTA_RESISTANCE command is set to YES, a delta network resistance model is used. The two views in Figure 208 are equivalent views of the same network.

Feedback

In the delta model, the resistance from node N1 to node N2 is still Rg, but the resistance from either node to node G is 1/3 Rg. The value of 1/3 Rg appears in the parasitic netlist.

This model requires negative resistance to achieve the intended result. Some simulators cannot handle negative resistance. In this case, set the `MOS_GATE_DELTA_RESISTANCE` command to `NO` and use the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command to use an alternate model that does not require negative resistance.

The MOS devices affected by this commands includes devices defined by the `ICV_LVS_DEVICE_TYPE_MOS` and `CALIBRE_LVS_DEVICE_TYPE_MOS` commands.

The `MOS_GATE_DELTA_RESISTANCE` command does not apply to X devices. To specify the gate resistance model for X devices, use the `MOS_GATE_DELTA_RESISTANCE_LAYERS` command.

See Also

- MOS_GATE_DELTA_RESISTANCE_LAYERS
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE

Feedback

MOS_GATE_DELTA_RESISTANCE_LAYERS

Specifies the database gate layers of MOS and X devices to extract using the delta model for gate resistance.

Syntax

```
MOS_GATE_DELTA_RESISTANCE_LAYERS: layer_list
```

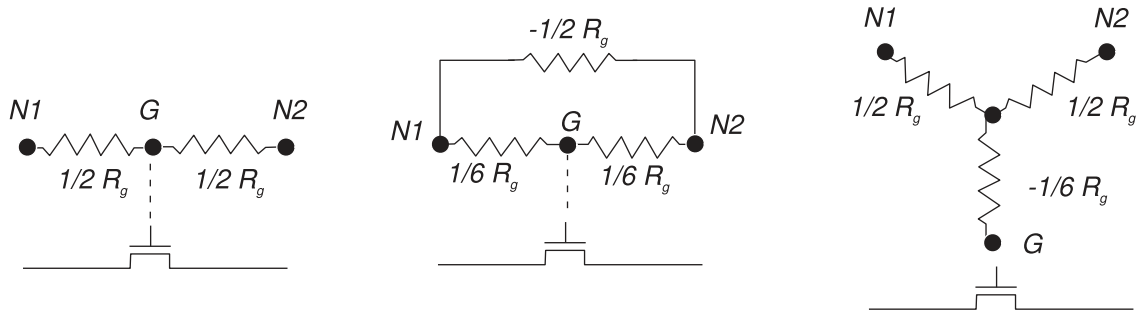
Arguments

Argument	Description
layer_list	The delta gate resistance model is used for all MOS and X devices whose gates are formed in these layers. Each layer must map to an ITF layer with LAYER_TYPE set to GATE. Space-delimited list of layers. Wildcards * and ! are allowed. Default: !* (no layers)

Description

Use the MOS_GATE_DELTA_RESISTANCE_LAYERS command in conjunction with the MOS_GATE_DELTA_RESISTANCE command to define whether the StarRC tool models MOS devices and X devices with the standard resistance model or the delta network resistance model. Figure 209 illustrates the two resistance models.

Figure 209 Standard Gate Resistance and Two Views of Delta Resistance Network



standard delta (view 1) delta (view 2)

Feedback

Use these commands in the StarRC command file as follows:

- To use the delta model only for specific MOS or X devices, list the gate terminal layers for those devices in the MOS_GATE_DELTA_RESISTANCE_LAYERS command. All other MOS and X devices use the standard resistance model.

```
MOS_GATE_DELTA_RESISTANCE: NO
MOS_GATE_DELTA_RESISTANCE_LAYERS: layer1 layer2 ...
```

- To use the delta model for all MOS devices plus specific X devices, list the gate terminal layers for the X devices in the MOS_GATE_DELTA_RESISTANCE_LAYERS command and set the MOS_GATE_DELTA_RESISTANCE command to YES. (The MOS_GATE_DELTA_RESISTANCE command does not apply to X devices.)

```
MOS_GATE_DELTA_RESISTANCE: YES
MOS_GATE_DELTA_RESISTANCE_LAYERS: layer1 layer2 ...
```

The MOS devices affected by these commands includes devices defined by the ICV_LVS_DEVICE_TYPE_MOS and CALIBRE_LVS_DEVICE_TYPE_MOS commands.

To apply delta network models to multifingered devices, the gate polysilicon must connect only to field polysilicon.

See Also

- MOS_GATE_DELTA_RESISTANCE
- ICV_LVS_DEVICE_TYPE_MOS
- CALIBRE_LVS_DEVICE_TYPE_MOS

Feedback

MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE

Extracts the gate resistance of MOS devices as an adjustable network without negative resistance.

Syntax

```
MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE: YES | NO_PG | NO
```

Arguments

Argument	Description
YES	Extracts gate resistance using a model that does not include negative resistance and allows scaling
NO_PG	Extracts gate resistance using a model without negative resistance and scales by excluding only those nets that are defined as POWER_NETS
NO (default)	Does not use this model

Description

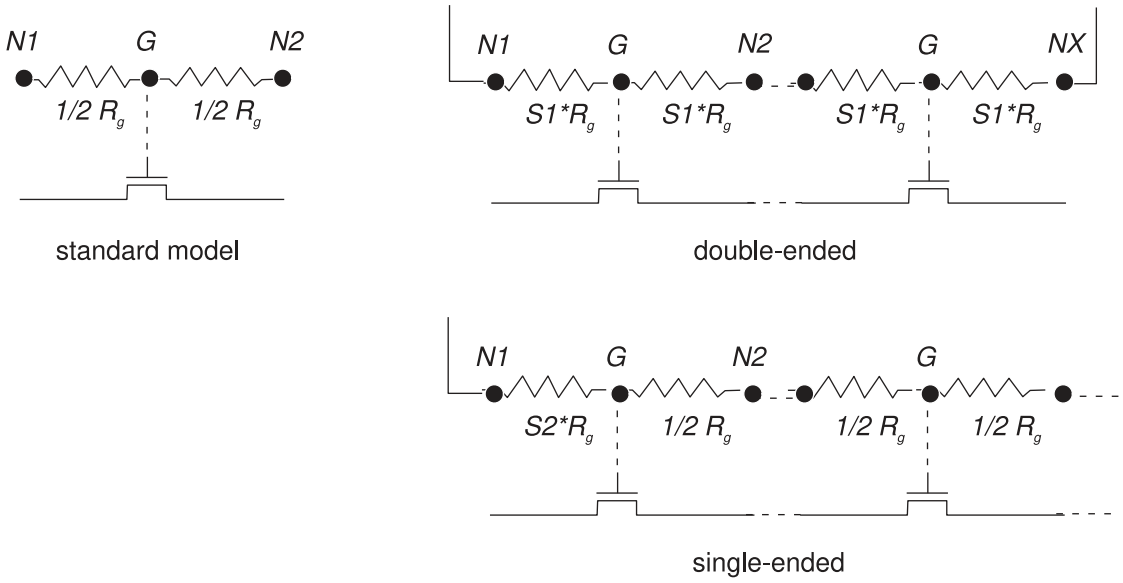
This command changes the gate resistance model for MOS devices to an adjustable model without negative resistance. Figure 210 shows the standard model and the two modes of the alternative model. Nodes N1 and N2 represent the ends of the gate polygon, while node G represents the ideal gate terminal location.

If devices are connected in series, node NX represents the end of the gate polygon of the last device. This alternative model has two design modes: double-ended (looped) mode in which nodes N1 and NX are both connected to a driver net, and single-ended mode in which only node N1 is connected to a driver net.

In the standard model, the resistance from node N1 to node N2 is Rg and the resistance from either node to node G is 1/2 Rg. The value of 1/2 Rg appears in the parasitic netlist.

Feedback

Figure 210 Nonnegative Gate Resistance Network Model



If the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE command is set to the following options:

- YES: The network resistance model shown in Figure 210 is used. S1 and S2 are scale factors as follows:
  - Scale factor S1 is for the double-ended mode. S1 is a multiplication factor applied to every resistance between node N1 and node G and between node N2 and node G for every transistor.

    Set this factor with the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE command. The default is 0.333.
  - Scale factor S2 is for the single-ended mode. S2 is a multiplication factor applied only to the resistance between node N1 (connected to the driver net) and node G. The resistance between node G and node N2 of this device and the node-to-gate resistance of all other devices in series are fixed at 1/2 Rg.

    Set this factor with the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE command. The default is 0.333.
- NO_PG: The tool excludes the nets that are identified as POWER_NETS during extraction. Extracts the non-negative delta resistor model on the gates of other nets only.

Feedback

See Also

- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_NETS
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_TAIL_COMMENT

Feedback

MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE

Scale factor for use with an alternate gate resistance model.

Syntax

```
MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE: factor_S1
```

Arguments

Argument	Description
factor_S1	Scale factor for the loop mode of the alternate resistance model Default: 0.333

Description

Specifies a scale factor to use for the loop mode (double-ended mode) of the alternate gate resistance model enabled with the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command.

See Also

- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE
- MOS_GATE_DELTA_RESISTANCE

Feedback

MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_NETS

Specifies a list of nets to extract the gate resistance of MOS devices as an adjustable network without negative resistance.

Syntax

```
MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_NETS: net_names
```

Arguments

Argument	Description
net_names	Lists the names of nets

Description

The command lists the names of the nets to be used by the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command. This helps to extract the gate resistance of MOS devices as an adjustable network without negative resistance for the specified nets only.

This command works when you set the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command to `YES` or `NO_PG` only . The asterisk (*) and negation (!) wildcards are accepted.

See Also

- MOS_GATE_DELTA_RESISTANCE

Feedback

MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE

Scale factor for use with an alternate gate resistance model.

Syntax

```
MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_SINGLE_SCALE: factor_S2
```

Arguments

Argument	Description
factor_S2	Scale factor for the loop mode of the alternate resistance model Default: 0.333

Description

Specifies a scale factor to use for the single-ended mode of the alternate gate resistance model enabled with the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` command.

See Also

- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_LOOP_SCALE
- MOS_GATE_DELTA_RESISTANCE

Feedback

MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_TAIL_COMMENT

Adds tail comments on gate resistors to write out in a SPF file for identifying single connected versus double connected gates.

Syntax

```
MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_TAIL_COMMENT: YES | NO
```

Arguments

Argument	Description
YES	Writes gate resistor connection type information as comments in the netlist
NO (default)	Disables writing of gate resistor connection type tail comments in a netlist

Description

The command controls whether the parasitic gate resistor connection type information is added to the netlist output file when you set the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE command to YES for MOS devices.

Note:

In the StarRC command file, you must set the REDUCTION: NO and NETLIST_TAIL_COMMENTs: YES commands before using the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_TAIL_COMMENT command.

When you set the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_TAIL_COMMENT command to YES, the StarRC tool includes the *$gate_drv* flag along with the gate resistor tail comments in the netlist. In the gate resistor tail comments,

- 1 is the value for single-end gate connection.
- 2 is the value for double-end gate connection.

This helps you to uniquely identify in the netlist whether the gates are connected on one end or connected on both ends.

Examples

Example 30 and Example 31 show the tail comments written in a SPF file when you set the MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE_TAIL_COMMENT command to YES for MOS devices.

Example 30 Data with double-ended case

```
R1_1 PMOS:GATE CLOCK:94 resStar R=1578.82 TC1=0.0015940974  $l=0.0810
 $w=0.0150 $lvl=182 $gate_drv=2
R1_2 PMOS:GATE CLOCK:100 resStar R=1578.82 TC1=0.0015940974  $l=0.0810
 $w=0.0150 $lvl=182 $gate_drv=2
```

Example 31 Data with single-ended case

```
R2_1 NMOS2:GATE INCLK:71 resStar R=2370.6 TC1=0.0015940977  $l=0.0810
 $w=0.0150 $lvl=182 $gate_drv=1
R2_2 NMOS:GATE INCLK:77 resStar R=1578.82 TC1=0.0015940974  $l=0.0810
 $w=0.0150 $lvl=182 $gate_drv=1
```

See Also

- MOS_GATE_DELTA_RESISTANCE
- NETLIST_TAIL_COMMENTS
- REDUCTION

Feedback

MULTI_PHYSICAL_PINS_PREFIX

Specifies whether to add a prefix to the pin names when a port has multiple physical pin names. Valid with Milkyway, NDM, and LEF/DEF designs.

Syntax

```
MULTI_PHYSICAL_PINS_PREFIX: YES | NO
```

Arguments

Argument	Description
YES	Add the SNPS_EEQ_ prefix to the pin names generated by the StarRC tool for ports with multiple physical pin names. Required to run hierarchical flows for ports with multiple physical pins.
NO (default)	Follow the standard naming behavior of the SHORT_PINS or SHORT_PINS_IN_CELLS commands.

Description

The MULTI_PHYSICAL_PINS_PREFIX command provides a method to distinguish two types of automatically-generated pin names. When ports have multiple physical pin names, the pins must be uniquely identified in the output parasitics file for proper back-annotation.

If you set the MULTI_PHYSICAL_PINS_PREFIX command to YES, the StarRC tool adds a prefix of SNPS_EEQ_ to an automatically-generated pin name for multiple electrically equivalent (EEQ) pins, as shown in Figure 211. The PrimeTime tool uses this information to distinguish between standard layout pins and EEQ pins.

Multiple Physical-Pins Hierarchical Flow

The multiple physical-pins hierarchical extraction flow enables block and full-chip analysis to overcome design complexity. In a flat extraction flow, the SHORT_PINS:YES and SHORT_PINS_IN_CELLS:* commands short the physical pins of top-level and skip-cell ports. In a hierarchical flow, the SHORT_PINS:MIXED and SHORT_PINS_IN_CELLS:!* commands create separate pin names for multiple physical pins with logical ports in a netlist to connect parasitics between various hierarchical extraction results (see Figure 212).

For information to create separate pin names for physical pins in LEF/DEF designs, see LEF/DEF Designs.

Figure 211 shows a hierarchical design example with the macro definitions.

Feedback

Figure 211 macro_pin Assigned to Two Pin Shapes In the First Group of Ports

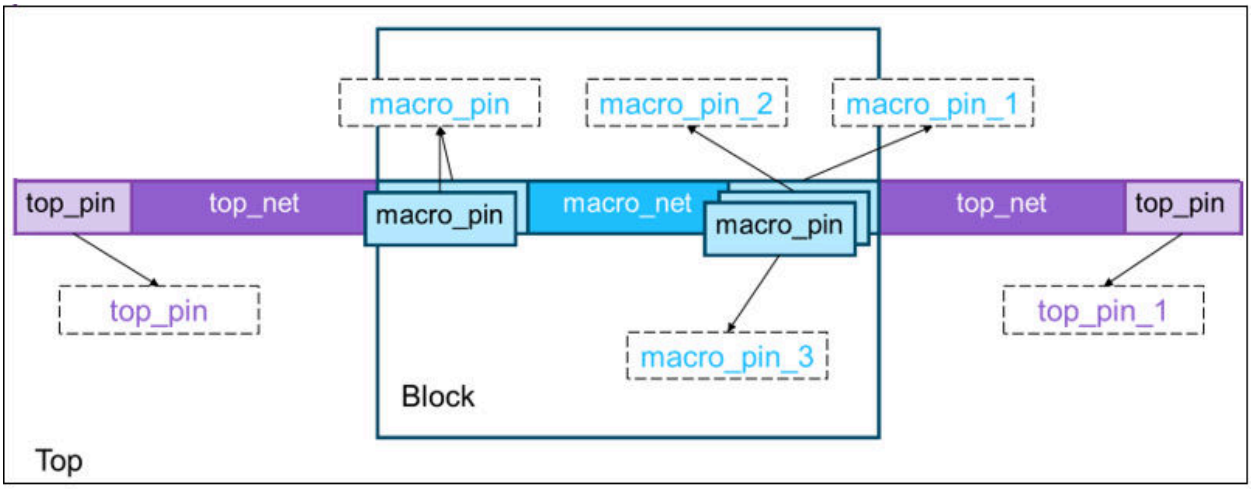
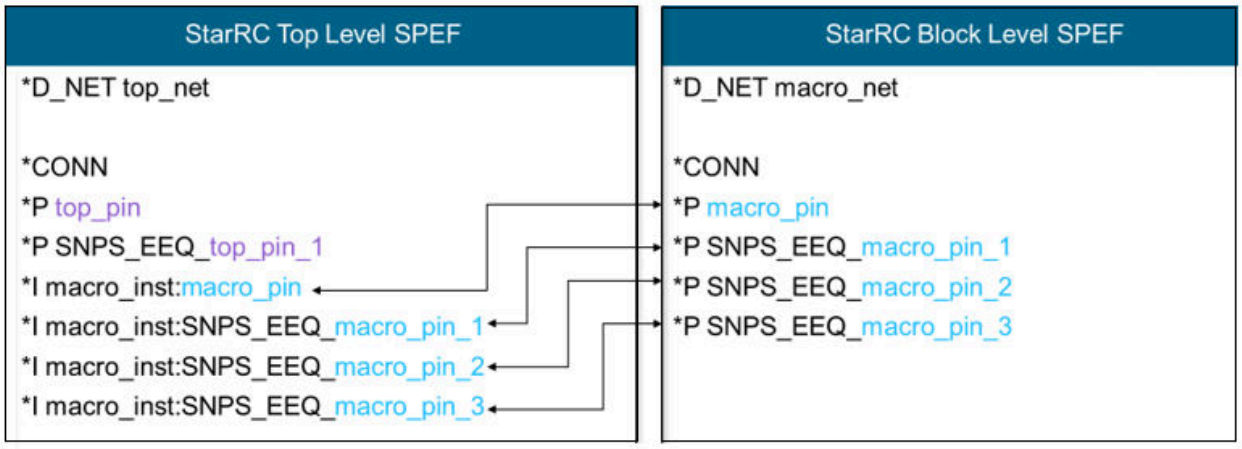


Figure 212 shows the result of the multiple physical-pins hierarchical extraction flow using the settings in Figure 211.

Figure 212 Prefix of SNPS_EEQ_ to PIN Names



Milkyway Designs

When extracting a block that contains ports with multiple physical pins as a top-level block or when extracting multiple instances of a block that contains ports with multiple physical pin, use the following StarRC commands:

```
MULTI_PHYSICAL_PINS_PREFIX: YES
MILKYWAY_DATABASE: design1
SHORT_PINS: MIXED
SHORT_PINS_IN_CELLS: !*
MILKYWAY_USE_CELL_PINS: YES
EXCLUDE_STDCELLS_FROM_SHORT_PINS_IN_CELLS: YES
```

Feedback

For more information, see MILKYWAY_USE_CELL_PINS.

NDM Designs

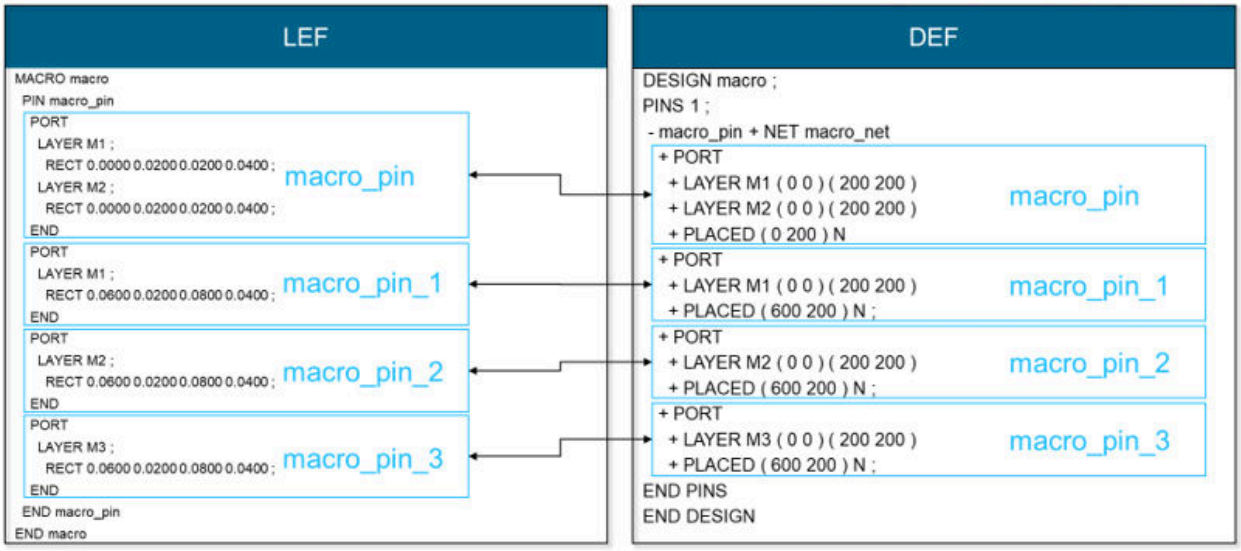
When extracting a block that contains ports with multiple physical pins as a top-level block or when extracting multiple instances of a block that contains ports with multiple physical pin, use the following StarRC commands:

```
MULTI_PHYSICAL_PINS_PREFIX: YES
NDM_DATABASE: design1
SHORT_PINS: MIXED
SHORT_PINS_IN_CELLS: !*
NDM_USE_DESIGN_PINS: YES
EXCLUDE_STDCELLS_FROM_SHORT_PINS_IN_CELLS: YES
```

For more information, see NDM_USE_DESIGN_PINS.

LEF/DEF Designs

In a LEF/DEF database, when you set the `MULTI_PHYSCIAL_PINS_PREFIX` command to `YES`, the StarRC tool creates a separate name for each group of pin shapes along with their associated pin names, as follows: the first group of ports gets the same name as the pin name, the second group of pin shapes gets the pin name with the suffix of _1 (PIN_1), the third group of pin shapes gets the pin name with the suffix of _2 (PIN_2), and so on. The group of pin shapes are recognized by the PORT syntax and pins are sequentially named in both LEF and DEF files, as shown in the following figure:



Feedback

You must set the commands as follows to set hierarchical extraction settings for top-level and block-level extraction, as shown in Table 73:

- Set the MULTI_PHYSCIAL_PINS_PREFIX and SHORT_PIN:MIXED commands to YES. This creates a separate name for each group of pin shapes with their associated pin name.
- Set the SHORT_PINS_IN_CELLS command to !* for the hierarchical extraction results to connect parasitics to the lowest level pins.
- Provide LEF files of the block level and set the DEF_USE_PINS to NO to use PIN definitions from the LEF files for block-level extraction.
- Use the LEF macro file to maintain consistency of the pins during the top-level extraction.

Table 73 Hierarchical Extraction Settings for Top-Level and Block-Level Extraction

StarRC Top-Level Extraction	StarRC Block-Level Extraction
LEF_FILE: tech.lef LEF_FILE: macro.lef TOP_DEF_FILE: top.def MULTI_PHYSICAL_PINS_PREFIX: YES SHORT_PINS: MIXED SHORT_PINS_IN_CELLS: !*	LEF_FILE: tech.lef LEF_FILE: macro.lef TOP_DEF_FILE: macro.def DEF_USE_PINS: NO MULTI_PHYSICAL_PINS_PREFIX: YES SHORT_PINS: MIXED SHORT_PINS_IN_CELLS: !***(optional, if not required to connect down to the next level)***

Examples

Port A has two terminals A and A_1, therefore the StarRC tool adds the SNPS_EEQ_ prefix to the name of terminal A_1. Port B has only a single terminal, therefore it does not have the prefix.

```
*D_NET netA 20.00027084
*CONN
*P SNPS_EEQ_A_1 O *C 1124.200 118.255 // $llx=1124.172 $lly=118.255
 $urx=1124.228 $ury=118.255 $lvl=6
*P A O *C 1124.088 0.454 // $llx=1124.060 $lly=0.445 $urx=1124.116
 $ury=0.462 $lvl=6
*P B O *C 1124.088 0.654 // $llx=1124.060 $lly=0.445 $urx=1124.116
 $ury=0.662 $lvl=8
```

Feedback

See Also

- SHORT_PINS
- SHORT_PINS_IN_CELLS
- EXCLUDE_STDCELLS_FROM_SHORT_PINS_IN_CELLS
- MILKYWAY_USE_CELL_PINS
- NDM_USE_DESIGN_PINS

Feedback

MULTIGATE_MODELS

Enables FinFET modeling.

Syntax

```
MULTIGATE_MODELS: YES | NO
```

Arguments

Argument	Description
YES	Enables FinFET modeling
NO (default)	Disables FinFET modeling

Description

To use FinFET models in the nxtgrd file, you must specify `MULTIGATE_MODELS: YES` in the StarRC command file. FinFET extraction requires an Ultra license. An Information message is generated if FinFET models are detected in the nxtgrd file but the `MULTIGATE_MODELS` command is not set to `YES`.

Examples

To use FinFET models, add the following command to the StarRC command file:

```
MULTIGATE_MODELS:YES
```

See Also

- MULTIGATE
- RPSQ_VS_SI_WIDTH_AND_LENGTH
- FinFET Modeling

Feedback

NDM_CELL_REPORT_FILE

Specifies the name of an optional file in which to write information about an IC Compiler II or a Fusion Compiler design.

Syntax

```
NDM_CELL_REPORT_FILE: ndm_report
```

Arguments

Argument	Description
ndm_report	Name of cell report file for an IC Compiler II or a Fusion Compiler design Default: none

Description

If you specify a file name with the `NDM_CELL_REPORT_FILE` command, the StarRC tool creates an optional report that contains information about cells in an IC Compiler II or a Fusion Compiler design. The report generated by the `NDM_CELL_REPORT_FILE` command contains the following information:

- A header that contains the StarRC version, the top cell name, and the date
- For each cell in the design, a section that includes the library name, cell name, cell version, database format, and library path (reference path)

Standard cells are reported first, followed by macro cells. The top-level cell is last. The cell version is the latest time stamp of the cell design file. The StarRC tool does not check for duplicate cell reference paths

Examples

An example of the cell report file is as follows:

```
                          StarRC (TM)
              Version M-2017.06 for linux 64 - June 5, 2017
...
Top Cell ............... routed_para_test
Data Format ............ NDM
Current Date ........... Mon Jun 5 02:20:19 2017

LIBNAME         CELLNAME  VERSION            FORMAT  LIBPATH
-------         --------  -------            ------  -------
cpu12b_cmos28   XOR3_C30  2015-117.12:56:43    NDM   /serv15/designs/...
cpu12b_cmos28   XOR3_C25  2015-117.12:56:43    NDM   /serv15/designs/...
...
```

Feedback

See Also

- NDM_DATABASE
- NDM_DESIGN_VIEW
- NDM_LAYOUT_VIEW
- NDM_EXPAND_HIERARCHICAL_CELLS
- NDM_CELL_REPORT_FILE
- NDM_SEARCH_PATH

Feedback

NDM_DATABASE

Specifies the name of a design library created by the Fusion Compiler or IC Compiler II tool.

Syntax

```
NDM_DATABASE: design_library
```

Arguments

Argument	Description
design_library	The name of the design library Default: none

Description

The NDM_DATABASE command specifies the name of the Fusion Compiler or IC Compiler II design library. This is the name that you use with the open_lib command in the Fusion Compiler or IC Compiler II tool.

The NDM_DATABASE command is mandatory for extraction based on an IC Compiler II or a Fusion Compiler flow. You must also use the StarRC BLOCK command to specify the top-level block name used for the open_block command in the Fusion Compiler or IC Compiler II tool.

Only gate-level designs are supported.

See Also

- BLOCK
- NDM_DESIGN_VIEW
- NDM_LAYOUT_VIEW
- NDM_EXPAND_HIERARCHICAL_CELLS
- NDM_CELL_REPORT_FILE
- NDM_SEARCH_PATH
- TRANSLATE_NDM_BLOCKAGE

Feedback

NDM_DESIGN_VIEW

A list of cells for which to use the DESIGN view instead of the FRAME view.

Syntax

```
NDM_DESIGN_VIEW: list_of_cells
```

Arguments

Argument	Description
list_of_cells	List of cells for which to use the DESIGN view during extraction, if available Default: none

Description

The argument of the NDM_DESIGN_VIEW command is a white-space-delimited list specifying cells for which the StarRC tool uses the DESIGN view instead of the FRAME view during extraction, if it is available.

You can specify this command multiple times in a single command file. The asterisk (*) and question mark (?) wildcard characters are acceptable.

Note:

This command applies to skip cells and their child cells only; the DESIGN view is always used for non-skip-cell masters.

For skip cells not on this list, the FRAME view represents the cell contents during extraction. The FRAME view typically contains all pin shapes and obstructions.

The StarRC tool attempts to translate the DESIGN view for each skip cell on this list. This view contains the actual physical layout, including nonroute layers. Specifying a cell with the NDM_DESIGN_VIEW command automatically includes all child cells. If a DESIGN view cannot be found for a cell contained in this list, the tool issues a warning and reverts to the FRAME view for that cell and all child cells.

Examples

```
NDM_DESIGN_VIEW: cell1 cell2 cell3
NDM_DESIGN_VIEW: cellA cellB cell? *C
NDM_DESIGN_VIEW: *
```

Feedback

See Also

- NDM_DATABASE
- NDM_CELL_REPORT_FILE
- NDM_EXPAND_HIERARCHICAL_CELLS
- NDM_LAYOUT_VIEW
- NDM_SEARCH_PATH

Feedback

NDM_EXPAND_HIERARCHICAL_CELLS

Enables flattening of cells in an IC Compiler II or a Fusion Compiler design library.

Syntax

```
NDM_EXPAND_HIERARCHICAL_CELLS: YES | NO
```

Arguments

Argument	Description
YES	Flattens cells of type NDM_DESIGN_TYPE_MACRO
NO (default)	Does not flatten cells

Description

If the NDM_EXPAND_HIERARCHICAL_CELLS command is set to YES, cells in an IC Compiler II or a Fusion Compiler design library whose type is NDM_DESIGN_TYPE_MACRO are flattened. Any other cell type remains a member of the list of skip cells specified by the SKIP_CELLS command.

The NDM_EXPAND_HIERARCHICAL_CELLS command automatically configures the skip cells list and takes precedence over the SKIP_CELLS command.

See Also

- NDM_DATABASE
- NDM_DESIGN_VIEW
- NDM_LAYOUT_VIEW
- NDM_SEARCH_PATH
- NDM_EXPAND_HIERARCHICAL_CELLS
- TRANSLATE_NDM_BLOCKAGE

Feedback

NDM_LAYOUT_VIEW

A list of skip cells for which to use the LAYOUT view in an IC Compiler II or a Fusion Compiler design.

Syntax

```
NDM_LAYOUT_VIEW: list_of_cells
```

Arguments

Argument	Description
list_of_cells	List of skip cells for which to use the LAYOUT view during extraction, if available. Other cell types are ignored. Default: none

Description

The argument of the `NDM_LAYOUT_VIEW` command is a white-space-delimited list specifying skip cells for which the StarRC tool should use the LAYOUT view during extraction, if it is available.

You can specify this command multiple times. The asterisk (`*`) and question mark (`?`) wildcard characters are acceptable.

For each specified skip cell, either the FRAME view or the DESIGN view must exist. The StarRC tool obtains connectivity information as follows, listed from highest to lowest priority:

- From the DESIGN view, if the view exists and the skip cell is specified in an `NDM_DESIGN_VIEW` command
- Otherwise, from the FRAME view

The rest of the skip cell content comes from one of the following sources, listed from highest to lowest priority:

- From a GDSII file if the skip cell is specified in the `GDS_FILE` command
- From the LAYOUT view, if the LAYOUT view exists and the cell is specified in the `NDM_LAYOUT_VIEW` command
- From the DESIGN view, if the DESIGN view exists and the cell is specified in the `NDM_DESIGN_VIEW` command
- Otherwise, from the FRAME view

Feedback

Examples

```
NDM_LAYOUT_VIEW: cell1 cell2 cell3
NDM_LAYOUT_VIEW: cellA cellB cell? *C
NDM_LAYOUT_VIEW: *
```

See Also

- NDM_DATABASE
- NDM_EXPAND_HIERARCHICAL_CELLS
- NDM_SEARCH_PATH
- NDM_DESIGN_VIEW

Feedback

NDM_REPORT_SCHEMA

Specifies whether to list the schema version for NDM format designs into the NDM cell report file.

Syntax

```
NDM_REPORT_SCHEMA: YES | NO
```

Arguments

Argument	Description
YES	Reports the NDM format schema in the NDM cell report file
NO (default)	Does not flatten cells

Description

If the NDM_REPORT_SCHEMA command is set to YES, the tool writes the schema version number into the NDM cell report file.

The cell report file is optional and must be specified with the NDM_CELL_REPORT_FILE command. Otherwise, the NDM_REPORT_SCHEMA command has no effect.

If enabled, the NDM schema appears in the report as follows:

- In the VERSION column, enclosed in square brackets immediately after the version.
- In the FILLS section, with one line per used FILL design, using the following format:

  ```
  library_name:cell_name.view_name schema_version
  ```

  For example:

  ```
  A7S.nlib:A7s__nwropt.FILL.design 1.255
  ```

See Also

- NDM_DATABASE
- NDM_CELL_REPORT_FILE

Feedback

NDM_SEARCH_PATH

Paths to search for the reference libraries of an IC Compiler II or a Fusion Compiler design library.

Syntax

```
NDM_SEARCH_PATH: design_path
```

Arguments

Argument	Description
design_path	The search path of the design library Default: none

Description

The `NDM_SEARCH_PATH` command is an optional command that specifies a list of paths to search for the reference libraries of the IC Compiler II or a Fusion Compiler design library named in the `NDM_DATABASE` command.

This command is the equivalent of the `search_path` command in the Fusion Compiler or IC Compiler II tool.

See Also

- NDM_DATABASE
- NDM_DESIGN_VIEW
- NDM_EXPAND_HIERARCHICAL_CELLS

Feedback

NDM_USE_DESIGN_PINS

Maintains consistency in pin names from the block-level and top-level extraction by using pins from the DESIGN view .

Syntax

```
NDM_USE_DESIGN_PINS: | NO | YES
```

Arguments

Argument	Description
NO (default)	Uses pins from the FRAM view.
YES	Uses pins preferably from the DESIGN view. If there are no pins in the DESIGN view, uses pins from the FRAME view.

Description

In an NDM database, pin names in the FRAM and DESIGN views might not be consistent. To avoid inconsistencies between pin names from block-level and top-level extractions in the multiple physical-pins hierarchical flow, set the NDM_USE_DESIGN_PINS command to YES to use pin names from DESIGN views preferentially. If the DESIGN view does not contain a pin, then pin names are taken from the FRAM view.

For more information, see Multiple Physical-Pins Hierarchical Flow.

See Also

- MULTI_PHYSICAL_PINS_PREFIX

Feedback

NDM_ZERO_SPACING_BLOCKAGE

Specifies how to treat zero spacing blockage regions in an NDM format design.

Syntax

```
NDM_ZERO_SPACING_BLOCKAGE: YES | NO
```

Arguments

Argument	Description
YES	Translate zero spacing blockages
NO (default)	Does not translate zero spacing blockages

Description

The NDM_ZERO_SPACING_BLOCKAGE command specifies how the StarRC tool translates zero spacing routing blockages created in the Fusion Compiler or IC Compiler II tool for an NDM format design.

- If you set the command to NO (the default), the StarRC tool ignores zero spacing blockages.
- If you set this command to YES, the StarRC tool treats a blockage region as a single polygon. A blockage region in a conductor layer is therefore treated as a large conductor.

Note:

The NDM_ZERO_SPACING_BLOCKAGE and NDM_ZERO_SPACING_BLOCKAGE_RATIO commands are valid for NDM format designs using an In-Design extraction flow in the Fusion Compiler or IC Compiler II tool. You can also use these commands in a StarRC extraction flow for the purpose of achieving compatibility with the Fusion Compiler or IC Compiler II extraction. Do not use these commands for signoff flows, because blockages do not represent the final design.

To improve the correlation between the capacitance extracted with the blockage and the final capacitance, set the NDM_ZERO_SPACING_BLOCKAGE_RATIO command to a density value that approximates the density of structures to be placed in the blockage region.

See Also

- NDM_ZERO_SPACING_BLOCKAGE_RATIO

Feedback

NDM_ZERO_SPACING_BLOCKAGE_RATIO

Specifies a density ratio to apply to zero spacing blockages in an NDM format design.

Syntax

```
NDM_ZERO_SPACING_BLOCKAGE_RATIO: ratio
```

Arguments

Argument	Description
ratio	A density value that approximates the density of structures to be placed in the blockage region Units: none Range: 0.0 (not inclusive) to 1.0 (inclusive) Default: 1.0

Description

The NDM_ZERO_SPACING_BLOCKAGE_RATIO command specifies the density ratio to apply to zero spacing blockages in an NDM format Fusion Compiler or IC Compiler II design. This command has an effect only if the NDM_ZERO_SPACING_BLOCKAGE command is set to YES.

Note:

The NDM_ZERO_SPACING_BLOCKAGE and NDM_ZERO_SPACING_BLOCKAGE_RATIO commands are valid for NDM format designs using an In-Design extraction flow in the Fusion Compiler or IC Compiler II tool. You can also use these commands in a StarRC extraction flow for the purpose of achieving compatibility with the Fusion Compiler or IC Compiler II extraction. Do not use these commands for signoff flows, because blockages do not represent the final design.

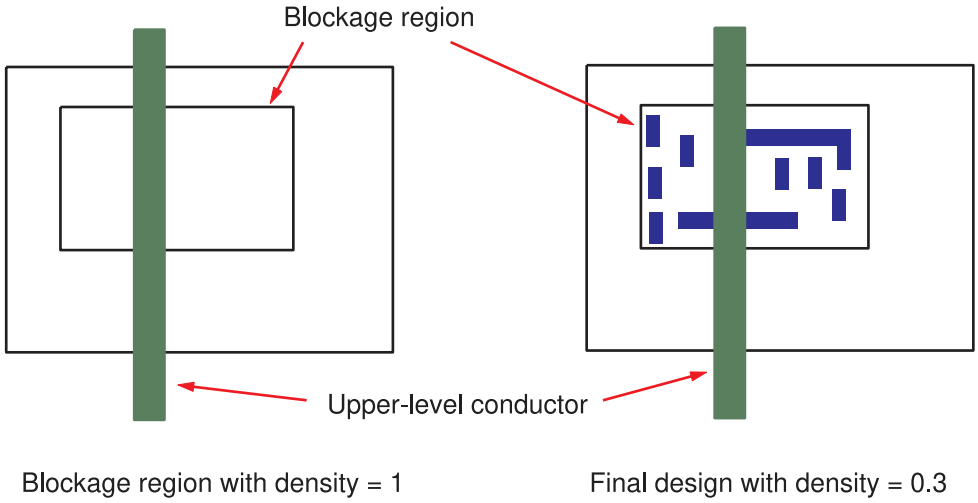
In the final design, the blockage region is replaced by polygons that do not occupy the full blockage area. Figure 213 illustrates an upper-level conductor passing over a blockage region before and after the final design is in place.

The extracted capacitance of the upper-level net in relation to the blockage region might be quite different after the final design is in place because the design shapes do not occupy the full blockage area.

To improve the correlation between the capacitance extracted with the blockage and the final capacitance, set the NDM_ZERO_SPACING_BLOCKAGE_RATIO command to a coverage density, where 1.0 indicates complete coverage of the blockage region. A value of 0.0 indicates no coverage; the value must be larger than 0.

Feedback

Figure 213 Zero Spacing Blockage



See Also

- NDM_ZERO_SPACING_BLOCKAGE

Feedback

NET_PREFIX

Specifies a name to match trivial ports with the specified net prefix.

Syntax

```
NET_PREFIX: string_name
```

Arguments

Argument	Description
string_name	Number specified in the StarRC command file

Description

A trivial port is defined as a port net of a skip cell (black-box cell) without the following information:

- Logical connectivity in the skip cell and without text (indicating that the net is a number from the LVS tool).
- Text, indicating that the net is identified by a number from the LVS tool.

If the LVS tool runset or rule deck defines a net prefix string for nets without text, you can then define this string through the `NET_PREFIX` command to match the trivial ports with the `NET_PREFIX`.

The `DELETE_TRIVIAL_INSTANCE_PORTS`, `EXPLODE_TRIVIAL_INSTANCE_PORTS`, and `REMOVE_TRIVIAL_INSTANCE_PORTS` commands operate only on a trivial port and remove the skip cell (black-box cell) ports that are not connected to devices.

The following example shows how the IC Validator runset tool defines a prefix on a net without text:

```
text_options(
    net_prefix = "N"
    );
```

The net without text is specified in the StarRC command file with a prefix N and a number to identify the tirvial port . From the example, `NET_PREFIX: N` identifies N6 as a trivial port of the nets if the port has no connectivity in the cell.

Feedback

NET_SEGMENT_CUT_LENGTH

Specifies the length of a cut segment. Valid only for transistor-level flows.

Syntax

```
NET_SEGMENT_CUT_LENGTH: cut_length
```

Arguments

Argument	Description
cut_length	Length of cut segment Units: microns Default: 20

Description

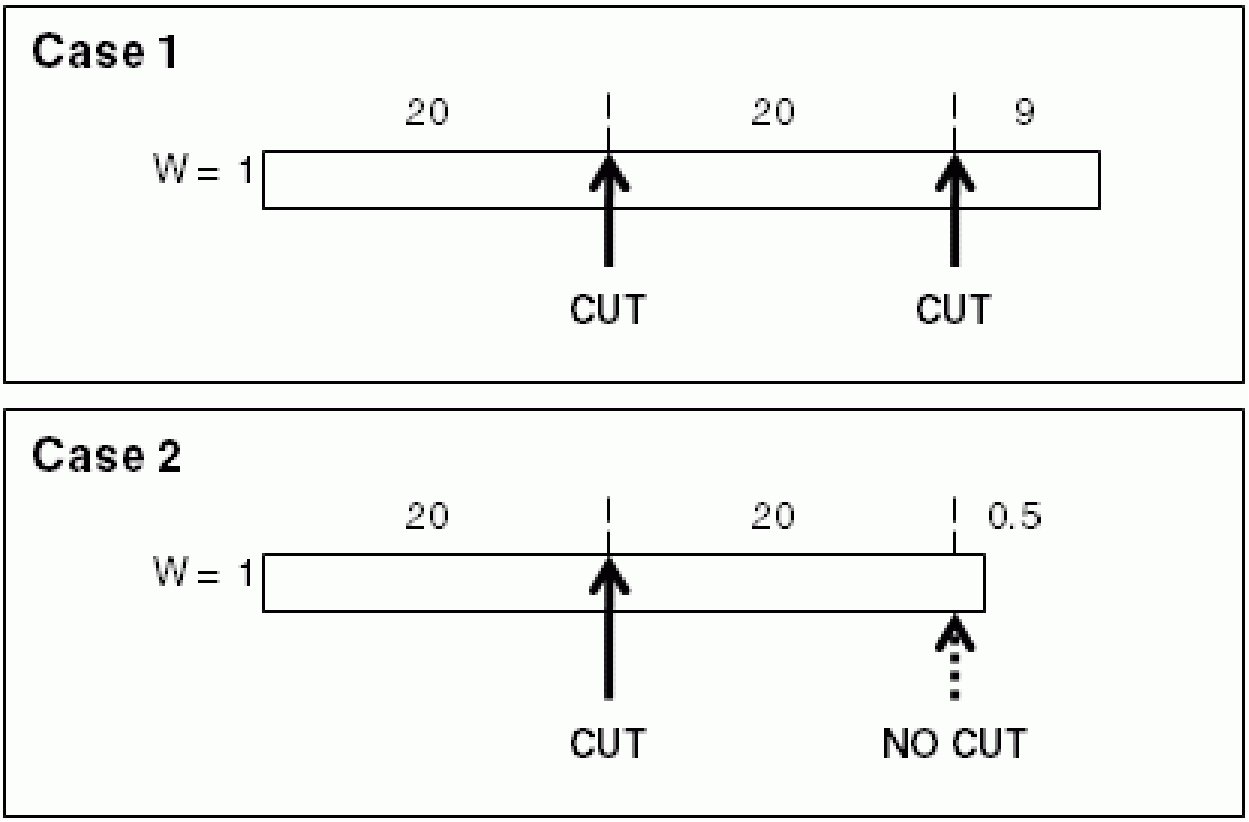
The StarRC tool cuts polygons of straight paths along their lengths. You can use the NET_SEGMENT_CUT_LENGTH command to specify the default segment length. The length of each resulting segment must be at least twice the width of the path. A cut is not made if it would result in a segment that is shorter than twice the width of the path.

If you enable netlist reduction with the REDUCTION command, then the additional nodes created by the NET_SEGMENT_CUT_LENGTH command are merged based on error control.

Figure 214 shows two paths that are one micron wide. The default segment cut length is 20 microns. In Case 1, the length of the last segment is 9 microns, which is acceptable because it is more than twice the width of the path. In Case 2, the length of the last

segment is 0.5 microns, which is less than twice the width. Therefore, the second cut is not made and the length of the last segment is 20.5 microns.

Figure 214 Cut Segments for NET_SEGMENT_CUT_LENGTH: 20



See Also

- REDUCTION
- conducting_layers

Feedback

NET_TYPE

Specifies the use of layout or schematic names during data selection.

Syntax

```
NET_TYPE: LAYOUT | SCHEMATIC
```

Arguments

Argument	Description
LAYOUT (default)	Specifies that the net names in a NETS: command are referenced to the layout
SCHEMATIC	Specifies that the net names in a NETS: command are referenced to the schematic

Description

Milkyway XTR (extraction) databases contain both layout names and cross-referenced schematic names. This command determines which set of names to use when looking up NETS and POWER_NETS during data selection.

This command is ignored if the MILKYWAY_EXTRACT_VIEW: NO (Hercules flow) or XREF: NO command is specified.

Note:

The NET_TYPE command identifies only the source of net names for selection in the NETS command. Reported net names are not affected.

See Also

- XREF
- CELL_TYPE
- MILKYWAY_EXTRACT_VIEW
- NETS
- POWER_NETS

Feedback

NETLIST_CAPACITANCE_UNIT

Specifies the units used for reporting capacitance values.

Syntax

```
NETLIST_CAPACITANCE_UNIT: cap_unit
```

Arguments

Argument	Description
cap_unit	The unit of capacitance for SPEF format Units: farads Default: 1e-15

Description

This command alters the units used for reporting capacitance values in both the header and the body of the output netlist. Applicable only for SPEF netlists.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_COMMENTED_PARAMS

Lists instance parameters in the netlist as comments. Valid only for transistor-level extraction.

Syntax

```
NETLIST_COMMENTED_PARAMS: YES | NO
```

Arguments

Argument	Description
YES	Lists instance parameters beginning with a '$' SPICE comment in the netlist
NO (default)	Does not list instance parameters in the netlist

Description

Specifies whether to generate instance parameters in the netlist that begins with a '$' SPICE comment. These parameters are from the layout spice or the property annotation files, which are the outputs generated by the LVS tool. Extra terminals ($BULK) and $.model are always included in the netlist.

Feedback

NETLIST_COMMENTS_FILE

Inserts the contents of specified files into the parasitic netlist as comments.

Syntax

```
NETLIST_COMMENTS_FILE: file1 file2 …
```

Arguments

Argument	Description
file1, file2, ...	File names whose contents are to be appended to the output netlist file

Description

Inserts the contents of specified files into the parasitic netlist as comments. This section begins after the netlist HEADER is printed. Each line from the file is inserted as is, prefixed by a comment string (`//` in SPEF format, `**` in all other formats). Empty lines are not included.

See Also

- NETLIST_FILE
- NETLIST_FORMAT

Feedback

NETLIST_COMPRESS

Specifies whether to compress a netlist generated from a GPD. Valid only in a GPD configuration file.

Syntax

```
NETLIST_COMPRESS: YES | NO
```

Arguments

Argument	Description
YES	Compresses the netlist
NO	Does not compress the netlist

Description

Use this command in a GPD configuration file to specify whether to compress a netlist created from the GPD.

See Also

- GPD
- The Parasitic Database or GPD

Feedback

NETLIST_COMPRESS_COMMAND

Specifies an executable for compression of an ASCII parasitic netlist.

Syntax

```
NETLIST_COMPRESS_COMMAND: utility [options]
```

Arguments

Argument	Description
utility	Executable to be run on the netlist

Description

The NETLIST_COMPRESS_COMMAND command provides a command for compressing an ASCII netlist during a run. Using the NETLIST_COMPRESS_COMMAND command to compress a netlist is faster than using the NETLIST_POSTPROCESS_COMMAND command for the same purpose.

Operations take place in real time. It is not possible to perform any operations on the netlist after it is saved. OA netlists cannot be processed; in this case, the StarRC tool issues a warning message.

The argument must be an executable utility program for file compression. If the specified executable is not in a directory specified by the PATH environment variable, you must specify the full path to the executable. The StarRC tool does not check for the existence of the executable. If the specified file is not found, the StarRC run exits with an error message.

The tool does not modify the netlist file name as a result of using this command.

This command can be used with the NETLIST_POSTPROCESS_COMMAND command to perform two operations on a netlist. Only one instance of each command is allowed in the StarRC command file. If both commands are present, the NETLIST_POSTPROCESS_COMMAND operation is performed first regardless of the order in the command file.

Examples

To compress the netlist with gzip, use the following command:

```
NETLIST_COMPRESS_COMMAND: /usr/local/bin/gzip
```

See Also

- NETLIST_POSTPROCESS_COMMAND

Feedback

NETLIST_CONNECT_OPENS

Creates a white-space-delimited list of nets to connect if found to be open in the input database.

Syntax

```
NETLIST_CONNECT_OPENS: netnames
```

Arguments

Argument	Description
netnames	Specifies the net names to be connected by a shorting resistor if the net is found to be open during extraction Default: all nets (*)

Description

By default, the StarRC tool identifies segments of an open net and connects them by inserting one or more shorting resistors. The tool detects opens by recognizing layout shapes that have the same net name or net number but are not physically connected.

Each segment of an open net is called a resistively connected group (RCG). Inserting resistors makes it possible for most timing analysis tools to calculate delays, even though one or more nets are not actually connected. Although the intended connectivity of the open net is not exactly known, the tool uses geometric proximity and layer compatibility to select nodes between which to insert the shorting resistors.

Shorting resistors have a resistance of 0.01 ohms and a width of 100 units to make them easy to recognize in the parasitic netlist. Settings of the REDUCTION and EXTRACTION commands might cause the coordinates of the shorting resistors to vary.

By default, the tool connects all open nets. You can control this behavior by using the NETLIST_CONNECT_OPENS command to specify which nets the tool should connect.

The asterisk (*), question mark (?), and exclamation mark (!) wildcards are acceptable. You can specify the NETLIST_CONNECT_OPENS command multiple times in a single command file; however, only the last command is honored.

Table 74 describes the behavior of the NETLIST_CONNECT_OPENS command for different combinations of input.

Table 74

Argument	NETLIST_CONNECT_OPENS command behavior
*	Connects all nets (default)

Feedback

Table 74 (Continued)

Argument	NETLIST_CONNECT_OPENS command behavior
* A B	Connects all nets
* !A	Connects all nets except net A
* !XY*	Connects all nets except nets with names beginning with XY
!*	Connects nothing
!* A	Connects only net A
!* !A	Connects nothing
!* XY*	Connects only nets with names beginning with XY
!* !XY*	Connects nothing
A !B C !D	Connects only nets A and C
!A	Connects nothing

Examples

In the following example, shorting resistors connect nets in resistively connected groups whose names do not begin with the strings ‘pwr’ or ‘gnd.’

```
NETLIST_CONNECT_OPENS: * !pwr* !gnd*

*RES
742 6:425 6:970 0.0100000
743 6:970 6:1445 0.0100000
```

See Also

- Opens Reports

Feedback

NETLIST_CONNECT_SECTION

Specifies whether the *I, *P, or *CONN sections are written to the output file.

Syntax

```
NETLIST_CONNECT_SECTION: YES | NO
```

Arguments

Argument	Description
YES (default)	Writes the *I, *P, or *CONN sections to the output file
NO	Does not write the *I, *P, or *CONN sections to the output file

Description

Applicable for all noncapacitor-only formats, including NETNAME. Setting this command to NO disables the generation of the information normally contained in the *|I and *|P or *CONN sections. This can reduce the netlist size significantly, but most delay calculators and static timing analysis tools require this information.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_COUPLE_UNSELECTED_NETS

Specifies whether to include the coupling capacitances of unselected nets in the netlist.

Syntax

```
NETLIST_COUPLE_UNSELECTED_NETS: IDEAL | COMPLETE | NO
```

Arguments

Argument	Description
IDEAL	Coupling capacitances from nets specified by the NETLIST_SELECT_NETS command to unselected nets are retained. Unselected nets do not have *|NET sections.
COMPLETE	Coupling capacitances to unselected nets are retained. Unselected nets have *|NET parasitic sections.
NO (default)	Coupling capacitances to unselected nets are not retained.

Description

This command specifies whether the netlist includes unselected nets (nets not specified by the NETLIST_SELECT_NETS command) or nets that are coupled to selected nets (nets specified by the command).

For coupled nets to be present in the output netlist, the extraction must be coupled by setting the EXTRACTION command to RC or C and the COUPLE_TO_GROUND command to NO.

Specify the netlist type for this option by using net names with the NETLIST_TYPE command. Coupling capacitances to unselected nets are retained if the NETLIST_TYPE:no_couple command is not set for the unselected nets to which the couplings exist. The NETLIST_TYPE:no_couple command overrides the NETLIST_COUPLE_UNSELECTED_NETS command for any unselected nets described in the NETLIST_TYPE command. If the NETLIST_TYPE: no_couple command is specified for unselected nets that are coupled to selected nets, neither the unselected nets nor their couplings to selected nets are retained.

See Also

- NETLIST_SELECT_NETS
- NETLIST_TYPE

Feedback

NETLIST_DELIMITER

Specifies the instance pin delimiter.

Syntax

```
NETLIST_DELIMITER: : | | | . | / | #
```

Arguments

Argument	Description
: (default)	Colon (:) character
|	Pipe (|) character
/	Slash (/) character
.	Period (.) character
#	Pound sign or hash (#) character

Description

Sets the instance pin delimiter to be printed in the output parasitic netlist.

Feedback

NETLIST_DEVICE_LOCATION_ORIENTATION

Writes device location information to the netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_DEVICE_LOCATION_ORIENTATION: YES | NO | COMMENT
```

Arguments

Argument	Description
YES	Device x-location, y-location, and orientation (angle) values are written in the instance section of the netlist
NO (default)	Location and orientation information is not written in the netlist
COMMENT	The location and orientation information is written to the netlist with a dollar sign ($) prefix for the parameter labels

Description

In transistor-level flows, you can set the NETLIST_DEVICE_LOCATION_ORIENTATION command to YES to write device location information into an SPF or NETNAME netlist. The extra information consists of the x and y locations and the angle. The angle is nonzero only for MOS devices.

To avoid potential conflict, ensure that the x and y locations, and the angle values (x/y/angle) or the $x/$y/$angle values with a dollar sign prefix that the StarRC tool additionally adds to the instance section (depending on the YES or COMMENT option) do not already exist in the LVS output.

Examples

The following example shows the netlist appearance when the command is set to NO:

```
MM1 10753:F40289 97802:F40290 10755:F40291 vgnd:F40288 MN ad=5.4p as=9.6p
pd=20.54u ps=40.96u l=0.18u w=20u
```

The following example shows the netlist appearance when the command is set to YES:

```
MM1 10753:F40289 97802:F40290 10755:F40291 vgnd:F40288 MN ad=5.4p as=9.6p
pd=20.54u ps=40.96u l=0.18u w=20u x=-1873.77 y=1789.68 angle=0
```

The following example shows the netlist appearance when the command is set to COMMENT:

```
MM1 10753:F40289 97802:F40290 10755:F40291 vgnd:F40288 MN ad=5.4p as=9.6p
pd=20.54u ps=40.96u l=0.18u w=20u $x=-1873.77 $y=1789.68 $angle=0
```

Feedback

See Also

- NETLIST_FORMAT

Feedback

NETLIST_FILE

Specifies the name of the file to which the output parasitic netlist is written.

Syntax

```
NETLIST_FILE: file_name
```

Arguments

Argument	Description
file_name	The generated output file. Default: *block_name*.spf, where *block_name* is the block specified by the BLOCK command

Description

The NETLIST_FILE command specifies a file name for the output netlist.

Table 75 shows how the NETLIST_FILE command affects the names of output netlists created using simultaneous multicorner (SMC) extraction.

Table 75 SMC Output Netlist File Names and Headers

NETLIST_FILE	SELECTED_CORNERS	File names	File headers (prefaced with ** for SPF files or // for SPEF files)
<*fname*>	<*corner1*>	<*fname*>.<*corner1*>	CORNER_NAME <*corner1*>
not used	<*corner1*>	<*block*>.spf.<*corner1*>	CORNER_NAME <*corner1*>
<*fname*>	<*corner1*> <*corner2*>	<*fname*>.<*corner1*> <*fname*>.<*corner2*>	CORNER_NAME <*corner1*> CORNER_NAME <*corner2*>
not used	<*corner1*> <*corner2*>	<*block*>.spf.<*corner1*> <*block*>.spf.<*corner2*>	CORNER_NAME <*corner1*> CORNER_NAME <*corner2*>
<*fname*>	<*c1*>:<*c2*>	<*fname*>.<*c1*>:<*c2*>	CORNER_NAME <*c1*>:<*2*>
not used	<*c1*>:<*c2*>	<*block*>.spf.<*c1*>:<*c2*>	CORNER_NAME <*c1*>:<*c2*>

The NETLIST_FILE command, in conjunction with the NETLIST_FORMAT command, affects whether data is stored in the GPD and whether a netlist is created, as shown in Table 76 for gate-level flows. Table 77 shows the same information for transistor-level flows.

Feedback

Table 76 Effect of Command Settings On Output Files and GPD for Gate-Level Flows

NETLIST_ FORMAT	NETLIST_ FILE	Data saved in GPD	Netlist format	Netlist file name
not used	not used	yes	not created	not created
SPEF	not used	yes	SPEF	<*block*>.spf
SPEF	<*fname*>	yes	SPEF	<*fname*>
SPF	not used	no	SPF	<*block*>.spf
SPF	<*fname*>	no	SPF	<*fname*>

Table 77 Command Settings For Output Files and GPD for Transistor-Level Flows

NETLIST_ FORMAT	NETLIST_ FILE	Data saved in GPD	Netlist format	Netlist file name
not used	not used	yes	not created	not created
SPF, NETNAME	not used	yes	SPF	<*block*>.spf
SPF, NETNAME	<*fname*>	yes	SPF	<*fname*>
OA	ignored	yes	not created	not created
STAR	not used	no	as specified	<*block*>.spf
STAR	<*fname*>	no	as specified	<*fname*>

Effect of gzip Compression on Netlist File Names

File compression with the gzip utility is commonly used to reduce netlist file size. Utilities that unzip files usually expect to find a file extension of .gz. Therefore, the file naming

convention is modified to accommodate this requirement. Specify the StarRC commands as follows:

- Use the `NETLIST_COMPRESS_COMMAND` command to specify the path to the gzip executable.
- Use the `NETLIST_FILE` command to specify a file name that ends in .gz. (Using the gzip utility does not automatically change the file name.) For example:

  ```
  NETLIST_FILE: run_23.spef.gz
  ```

In SMC extraction, the corner name is inserted before the .gz file extension, as follows:

```
NETLIST_FILE: run_23.spef.<corner1>.gz
```

See Also

- NETLIST_FORMAT
- GPD
- NETLIST_COMPRESS_COMMAND
- The Parasitic Database or GPD

Feedback

NETLIST_FORMAT

Defines the format of the output parasitic netlist.

Syntax

```
NETLIST_FORMAT: SPF | SPEF | OA | NETNAME | STAR | PARAMETERIZED_SPICE
```

Arguments

Argument	Description
SPF	Standard Parasitic Format. Supports only EXTRACTION: RC with COUPLE_TO_GROUND: YES | NO. Supports coupling capacitors.
SPEF	Flexible and compact. All names are mapped internally, reducing netlist size. SPEF prints the D_NET (detailed parasitics) net type in the netlist.
OA	Transistor-level extraction only. Creates an OpenAccess view.
NETNAME	Transistor-level extraction only. Formats internal node names as *netname*:1, *netname*:2, and so on .
STAR	Transistor-level extraction only. A compact format that uses SPICE-like subnode naming conventions.
PARAMETERIZED_SPICE	A parameterized SPICE netlist that enables Monte Carlo analysis by downstream simulators.

Description

The NETLIST_FORMAT command specifies the format for the output netlist.

The following behavior applies to all flows:

- The StarRC tool saves parasitics in the GPD by default.
- When a GPD is created, the StarRC tool does not create an ASCII netlist at the time of the extraction run unless you include the NETLIST_FORMAT and NETLIST_FILE commands in the command file.
- You can create a SPEF netlist from an existing GPD directory by using the StarXtract -convert_gpd_to_spef command.
- You can create an SPF netlist from an existing GPD directory by using the StarXtract -convert_gpd_to_spf command.

Feedback

The following behavior applies to transistor-level flows:

- If a GPD is not created due to the use of an unsupported command, you can create a new netlist from a completed transistor-level extraction run by using the `StarXtract -cleanN` command. If the initial run generated a netlist in the `SPF`, `OA`, or `NETNAME` formats, you can create a new netlist in any format. However, if the initial run generated a SPEF netlist or did not generate any netlist, you can create only SPEF netlists with the `StarXtract -cleanN` command.
- If a GPD is created, the `-cleanN` option is not allowed.

The `NETLIST_FORMAT` command, in conjunction with other commands, affects whether data is stored in the GPD and whether a netlist is created, as shown in Table 78 for gate-level flows. Table 79 shows the same information for transistor-level flows.

Table 78 Commands For Output Files and GPD for Gate-Level Flows

NETLIST_ FORMAT	NETLIST_ FILE	Data saved in GPD	Netlist format	Netlist file name
not used	not used	yes	not created	not created
SPEF	not used	yes	SPEF	<*block*>.spf
SPEF	<*fname*>	yes	SPEF	<*fname*>
SPF	not used	no	SPF	<*block*>.spf
SPF	<*fname*>	no	SPF	<*fname*>

Table 79 Command Settings For Output Files and GPD for Transistor-Level Flows

NETLIST_ FORMAT	NETLIST_ FILE	Data saved in GPD	Netlist format	Netlist file name
not used	not used	yes	not created	not created
SPF, NETNAME	not used	yes	SPF	<*block*>.spf
SPF, NETNAME	<*fname*>	yes	SPF	<*fname*>
OA	ignored	yes	not created	not created
STAR	not used	yes	as specified	<*block*>.spf

Feedback

Table 79 Command Settings For Output Files and GPD for Transistor-Level Flows (Continued)

NETLIST_ FORMAT	NETLIST_ FILE	Data saved in GPD	Netlist format	Netlist file name
STAR	<*fname*>	yes	as specified	<*fname*>

Note:

When you use the `OA_MULTI_OUTPUT` command along with the `NETLIST_FORMAT:OA` command, the tool honors the setting specified in the `NETLIST_FILE` command and uses the format of the `OA_MULTI_OUTPUT: SPF | SPEF | STAR` command to create an ASCII netlist output.

Parameterized Netlists for Wire Variation Analysis

You can create a special netlist that enables variation in downstream simulation tools by setting the `NETLIST_FORMAT` command to `PARAMETERIZED_SPICE`. The output consists of the following files:

- A SPICE netlist that contains equations for parasitic resistances and capacitances
- A side file that contains a template for the coefficients that must be defined to calculate the parasitics.

To use this option, the following conditions are required:

- A flow that enables saving data in a GPD.
- Simultaneous multicorner extraction with at least 2 corners.
- No temperature sensitivity analysis.
-

A parameterized netlist includes an identification line in the header, as follows:

```
**FORMAT PARAMETERIZED_SPICE
```

The netlist contains parameters that provide scaling for individual corner values per layer. The scaling parameter names use one of the following formats:

```
<corner-name>_<element-type>_<layer-id>
<corner-name>_<element-type>_<victim-layer-id>_<aggressor-layer-id>
```

Table 80 describes the syntax items in these formats.

Feedback

Table 80 SPICE Netlist Parameter Naming Format

Syntax item	Description
<corner-name>	The name of a selected corner in the corners file used for the simultaneous multicorner extraction run
<element-type>	RR (routing layer resistance) RV (via layer resistance) CG (ground capacitance) CC (coupling capacitance)
<layer-id>	For RR, RV, and CG elements: the layer ID
<vic-layer-id>_<aggr-layer-id>	For CC elements: the victim layer ID and the aggressor layer ID

For example, a scaling parameter named CWORST_RR_127 applies to routing layer resistors on layer 127 for corner CWORST. A scaling parameter named FAST_CC_12_15 applies to coupling capacitors between layers 12 and 15 for corner FAST.

The parameter names appear in the netlist as follows:

<element-label> <node1> <node2> ‘V_{TYP} * (1 + A*P_A + B*P_B + C*P_C + ...)’

The element label (R, Rv, Cc, or Cg followed by a numeric index), node1, and node2 are written according to the practices of a standard netlist. However, in a parameterized netlist, the element value is represented by an equation with the following components:

- The symbols P_A, P_B, and so on are replaced by the names (not values) of scaling parameters.
- The symbols A, B, and so on are replaced by coefficients determined by the StarRC tool based on the extraction results for the corners.
- The value V_{TYP} is the resistance or capacitance (as appropriate for the element type) for the first selected corner, which is usually a nominal or typical value.

For example, a parameterized netlist might contain the following lines:

```
Cg188_1 net45 0 '4.55126e-17*( 1 + 0*CWORST_CG_12 )'
R2550_1 net89 net92 '1250.0*( 1 - 0.0235567*CWORST_RR_12 )'
```

The side file, which is named *block*..wire_variation_params, contains the values of the scaling parameters in lines such as the following:

```
.param CWORST_CG_10 = 0.0
.param CWORST_CG_11 = 0.0
...
.param CWORST_RR_12 = 1.0
```

```
.param CWORST_RR_13 = 1.0
...
```

You can include the parameterized netlist and the side file in a SPICE run script by using SPICE `.include` statements. Perform further analysis as follows:

- Set all of the scaling parameters to 0.0 to obtain a resistance or capacitance value equal to the V_{TYP} value.
- Set all of the scaling parameters that contain a specific corner name to 1.0 to obtain a resistance or capacitance equal to the extracted value for that corner.
- Modify the values as needed to perform variation analysis. For example, you can use Monte Carlo analysis to apply random variations to the scaling parameters.

See Also

- NETLIST_FILE
- GPD
- The Parasitic Database or GPD

Feedback

NETLIST_GROUND_NODE_NAME

Defines the net name used when reporting the capacitance with respect either to noncritical material or to a SUBSTRATE layer in the ITF file. Valid only for transistor-level extraction.

Syntax

```
NETLIST_GROUND_NODE_NAME: net_name
```

Arguments

Argument	Description
net_name	The name of the net to which the capacitance is to be lumped Default: 0

Description

This command is not valid with SPEF format netlists, because the ground node is not included in a SPEF output file.

A reference to node 0 in the output netlist is the location where all noncritical extracted materials are lumped. This includes coupling to ideal ground or a SUBSTRATE layer. The entry for node 0 is the SPICE ground.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_HIER_PROBE_NODES

Specifies whether the net hierarchy must be reported in the RC netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_HIER_PROBE_NODES: YES | NO
```

Arguments

Argument	Description
`NO` (default)	Does not report the hierarchy in the output
`YES`	Reports the hierarchy information *cell_inst*:*text_label* in the RC netlist output

Description

This command specifies whether the net hierarchy must be reported in the RC netlist.

Examples

```
**|OI (cell_inst : text_label  cell_inst text_label
Z 0 x_coord y_coord
*| NET SUM0 0.0128485PF
**|OI ($1I1:ProbeA1 $1I1 ProbeA1 Z 0 459.5 34.5)
R16  $1I1:ProbeA1 SUM0 1.19335 $l = 38.495 $w = 2 $lvl = 1
```

The text `$1I1:ProbeA1` is inserted into the output when the `NETLIST_HIER_PROBE_NODES` command is set to `YES`.

Feedback

NETLIST_IDEAL_SPICE_FILE

Generates an ideal SPICE netlist for use with simulation tools. Valid only for transistor-level extraction.

Syntax

```
NETLIST_IDEAL_SPICE_FILE: file_name
```

Arguments

Argument	Description
file_name	Ideal SPICE netlist file name

Description

This command creates an ideal SPICE netlist for use with simulation tools.

The SPICE netlist stops at skip cell boundaries. Skip cells and device .SUBCKT statements are included in the netlist as comments to indicate port ordering. For skip cell extractions, use the SPICE_SUBCKT_FILE command to specify a file that contains the port ordering information for subcircuit ports.

In the generated file, the top-level .SUBCKT and .ENDS statements are commented out and must be edited before use. Check the port list and other contents against your requirements before using the file in a simulation.

The combination of files specified by the NETLIST_IDEAL_SPICE_FILE and SPICE_SUBCKT_FILE commands provides a complete device-level ideal SPICE netlist for use with simulation tools.

The file contents are also affected by the NETLIST_PASSIVE_PARAMS, NETLIST_MAX_LINE, NETLIST_IDEAL_SPICE_TYPE, NETLIST_IDEAL_SPICE_HIER, and SUPPORT_DIFFERENT_PORTNAME_NETNAME commands.

See Also

- NETLIST_IDEAL_SPICE_HIER
- NETLIST_IDEAL_SPICE_TYPE
- NETLIST_MAX_LINE
- NETLIST_PASSIVE_PARAMS
- SPICE_SUBCKT_FILE
- SUPPORT_DIFFERENT_PORTNAME_NETNAME

Feedback

NETLIST_IDEAL_SPICE_HIER

Specifies whether to preserve the original hierarchy when generating the ideal SPICE netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_IDEAL_SPICE_HIER: YES | NO
```

Arguments

Argument	Description
YES	Preserves the original netlist or layout hierarchy
NO (default)	Flattens the ideal netlist

Description

The NETLIST_IDEAL_SPICE_HIER command controls whether to preserve the original hierarchy when generating the file specified in the NETLIST_IDEAL_SPICE_FILE command.

When the CALIBRE_PDBA_FILE or ICV_ANNOTATION_FILE commands are used with the NETLIST_IDEAL_SPICE_HIER command, the StarRC output does not include the DFM properties in a separate device property file.

If the CALIBRE_PDBA_FILE command is used with the NETLIST_IDEAL_SPICE_HIER: YES command, the CALIBRE_PDBA_FILE command is ignored and the layout netlist hierarchy is preserved.

See Also

- CALIBRE_PDBA_FILE
- NETLIST_IDEAL_SPICE_FILE
- NETLIST_IDEAL_SPICE_TYPE

Feedback

NETLIST_IDEAL_SPICE_TYPE

Specifies whether to generate a layout- or schematic-based ideal SPICE file. Valid only for transistor-level extraction.

Syntax

```
NETLIST_IDEAL_SPICE_TYPE: SCHEMATIC | LAYOUT
```

Arguments

Argument	Description
`SCHEMATIC`	Uses schematic net names
`LAYOUT` (default)	Uses layout net names

Description

The default for `XREF:NO` is `LAYOUT`. The default for all other `XREF` values is `SCHEMATIC`.

See Also

- NETLIST_IDEAL_SPICE_FILE

Feedback

NETLIST_INCREMENTAL

Specifies whether to generate a full netlist or an incremental netlist during ECO extraction.

Syntax

```
NETLIST_INCREMENTAL: YES | NO
```

Arguments

Argument	Description
YES	Generates an incremental netlist if extraction performance is improved by doing so
NO (default)	Generates a full netlist regardless of the number of ECO changes detected

Description

ECO extraction is the technique of performing extraction only on parts of a design that are different from a reference design. This capability allows efficient evaluation of engineering change orders.

During ECO extraction, the StarRC tool evaluates the percentage of ECO-affected nets and extracts only those nets if runtime would be reduced by doing so. The full-chip parasitics are saved in the GPD.

A SPEF netlist is generated only if the command file contains the NETLIST_FILE and NETLIST_FORMAT commands. If the NETLIST_INCREMENTAL command is set to NO, the netlist is a full netlist. If the command is set to YES, the netlist is an incremental netlist.

See Also

- ECO_MODE
- Chapter 5, ECO Extraction

Feedback

NETLIST_INPUT_DRIVERS

Identifies the driving cell models in the SPEF netlist format only for each input instance pin with the optional *D statement.

Syntax

```
NETLIST_INPUT_DRIVERS: YES | NO
```

Arguments

Argument	Description
YES	Prints the driving cell model name of input instance pin in the netlist
NO (default)	Does not enable the command function

Description

A pin (*I) can be input, output, or bidirectional. Many static timing tools do not require this information for the input pins, because the loading cap for the net is provided. The StarRC tool does not print the *D statements for the inputs by default. Use this option to print the models for the input instance pins.

This command has an effect only if you specify the NETLIST_FORMAT: SPEF command.

See Also

- NETLIST_FORMAT
- NETLIST_OUTPUT_DRIVERS

Feedback

NETLIST_INSTANCE_SECTION

Specifies whether the instance section is included in the output netlist.

Syntax

```
NETLIST_INSTANCE_SECTION: YES | NO | SELECTED
```

Arguments

Argument	Description
YES	Includes all instances
NO (default for gate-level extraction)	Does not generate the instance section
SELECTED (default for transistor-level extraction)	Generates the instance section from the connected nets group that is the combination of the NETS and NETLIST_SELECT_NETS commands

Description

The NETLIST_INSTANCE_SECTION command is valid only when the NETLIST_FORMAT command is set to SPF, NETNAME, or OA.

Because .SUBCKT statements must be consistent with the instance section, they are also written to the netlist.

If the SPICE_SUBCKT_FILE command is used, the .SUBCKT line lists all ports, including ports that are not attached to selected nets. Otherwise, only the nets that are extracted (due to NETS or POWER_NETS command settings) are included in the .SUBCKT line.

Examples

The following examples describe the effects of different combinations of the NETS, XREF, NETLIST_INSTANCE_SECTION, and NETLIST_SELECT_NETS commands.

Example 1

```
NETS: selected
XREF:NO|YES|NETS
NETLIST_SELECT_NETS:*
```

If the NETLIST_INSTANCE_SECTION command is set to NO, the netlist does not contain an instance section.

If the NETLIST_INSTANCE_SECTION command is set to YES or SELECTED, the instance section contains devices attached to extracted nets.

Feedback

Example 2

```
NETS: selected
XREF:COMPLETE
NETLIST_SELECT_NETS:*
```

If the NETLIST_INSTANCE_SECTION command is set to NO, the netlist does not contain an instance section.

If the NETLIST_INSTANCE_SECTION command is set to YES or SELECTED, all schematic devices are included in the instance section. Nets that are not extracted are ideal nets.

Example 3

```
NETS: *
XREF:NO|YES|NETS
NETLIST_SELECT_NETS:selected
```

If the NETLIST_INSTANCE_SECTION command is set to NO, the netlist does not contain an instance section.

If the NETLIST_INSTANCE_SECTION command is set to YES, all devices are included.

If the NETLIST_INSTANCE_SECTION command is set to SELECTED, only devices attached to selected nets are included.

See Also

- NETLIST_FORMAT
- NETLIST_SELECT_NETS
- NETLIST_COUPLE_UNSELECTED_NETS
- NETS
- POWER_NETS

Feedback

NETLIST_LOCATION_TRANSFORMS

Writes location transformation information for automatically detected macros into the GPD..

Syntax

```
NETLIST_LOCATION_TRANSFORMS: YES | NO
```

Arguments

Argument	Description
YES	Writes location transformation factors into the GPD
NO (default)	Disables saving of location transformation factors

Description

Extraction and timing analysis tools must understand node and pin locations with respect to both local coordinates (within a block or macro) and global coordinates. In addition to being placed at one or more locations within a larger design, macros might be flipped or rotated.

Set the NETLIST_LOCATION_TRANSFORMS command to YES to save offset, rotation, and flip information into the GPD for automatically detected macros.

To save location transforms into the GPD for other cells, use the NETLIST_LOCATION_TRANSFORMS_ADDITIONAL_CELLS command.

If you create a Standard Parasitic Exchange Format (SPEF) output file from a GPD that contains transform factors, the SPEF file also contains the transform factors.

For LEF/DEF flows, the NETLIST_LOCATION_TRANSFORMS command saves transform information only for cells that have a DEF file listed in the MACRO_DEF_FILE command.

See Also

- GPD
- MACRO_DEF_FILE
- NETLIST_LOCATION_TRANSFORMS_ADDITIONAL_CELLS
- The Parasitic Database or GPD

Feedback

NETLIST_LOCATION_TRANSFORMS_ADDITIONAL_CELLS

Writes location transformation information for specific cells into the GPD.

Syntax

```
NETLIST_LOCATION_TRANSFORMS_ADDITIONAL_CELLS: cell1, cell2, ...
```

Arguments

Argument	Description
cell1, *cell2*, ...	Cells for which to write location transformation factors into the GPD

Description

The `NETLIST_LOCATION_TRANSFORMS:YES` command saves offset, rotation, and flip information into the GPD for macro cells that are automatically detected by the StarRC tool (macros that have a macro DEF file).

To save location transforms into the GPD for other cells, specify the cell names as arguments for the `NETLIST_LOCATION_TRANSFORMS_ADDITIONAL_CELLS` command. To use this command, you must also set the `NETLIST_LOCATION_TRANSFORMS` command to `YES`.

See Also

- GPD
- MACRO_DEF_FILE
- NETLIST_LOCATION_TRANSFORMS
- The Parasitic Database or GPD

Feedback

NETLIST_LOGICAL_TYPE

Sets a value to be printed in the `SPEF DESIGN_FLOW NETLIST_TYPE` *`val`* header.

Syntax

```
NETLIST_LOGICAL_TYPE: VERILOG | VHDL87 | VHDL93 | EDIF
```

Arguments

Argument	Description
`VERILOG`	Verilog is the naming convention
`VHDL87`	VHDL87 is the naming convention
`VHDL93`	VHDL93 is the naming convention
`EDIF`	EDIF is the naming convention

Description

This command specifies which naming convention is used in the creation of the SPEF netlist. The value is written in the `SPEF DESIGN_FLOW NETLIST_TYPE` header.

This command is not required by the StarRC tool, but might be necessary for follow-on tools to read the output netlist.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_MAX_LINE

Maximum number of characters to write on each netlist line.

Syntax

```
NETLIST_MAX_LINE: no_of_chars
```

Arguments

Argument	Description
no_of_chars	Maximum number of characters allowed on each netlist line Default: none

Description

This command applies to SPF netlists. The default is to place no limit on the number of characters in a line.

When writing a netlist, the StarRC tool limits line length in the file to the specified number of characters. If needed, the tool continues the text on the following line and writes the “+” continuation tag at the beginning of the second and subsequent lines.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_MERGE_SHORTED_PORTS

Removes 0.001-ohm node-sharing resistors and merges node names in the netlist to reduce the file size. Valid only for transistor-level extraction.

Syntax

```
NETLIST_MERGE_SHORTED_PORTS: YES | NO
```

Arguments

Argument	Description
YES	Instance ports connected by node sharing resistors are replaced by one node in the group
NO (default)	Instance nodes are unique and might be connected by node sharing resistors (if there are no physical parasitic resistors)

Description

If the NETLIST_MERGE_SHORTED_PORTS command is set to YES, whenever multiple port nodes for a net are connected together by node-sharing shorting resistors, the StarRC tool chooses one node randomly from the group to represent all nodes. The tool uses this node to replace every node in the group for every electrical element in the netlist including parasitic elements, elements in the instance section, and *|I occurrences in DSPF.

Examples

```
NETLIST_MERGE_SHORTED_PORTS: YES
```

Feedback

NETLIST_MINCAP_THRESHOLD

Sets the minimum capacitance allowed in the RC section of the netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_MINCAP_THRESHOLD: capacitance_value
```

Arguments

Argument	Description
capacitance_value	The smallest capacitance to be allowed without merging with another capacitance Units: farad (F) Default: 0.0

Description

Any capacitance below this threshold is merged with another smaller capacitance or larger capacitance in a given net. This is applicable for both coupling and grounded capacitance. The capacitance value cannot be less than 0 (zero).

Note:

Capacitance that is below the threshold can remain in the netlist.

When the `NETLIST_MINCAP_THRESHOLD` and `COUPLING_ABS_THRESHOLD` commands are both specified, the StarRC tool applies the `COUPLING_ABS_THRESHOLD` command first.

Examples

This sets the threshold level at 1 fF.

```
NETLIST_MINCAP_THRESHOLD: 1e-15
```

See Also

- COUPLING_ABS_THRESHOLD
- NETLIST_TOTALCAP_THRESHOLD

Feedback

NETLIST_MINRES_HANDLING

Specifies how a resistor is handled if it is less than or equal to the specified threshold. Valid only for transistor-level extraction.

Syntax

```
NETLIST_MINRES_HANDLING: SHORT | MERGE
```

Arguments

Argument	Description
SHORT	Does not preserve point-to-point resistance
MERGE (default)	Merges the resistor with a neighboring resistor if it is in series and is smaller than the threshold

Description

This command specifies how a resistor is handled if it is less than or equal to the specified threshold in the NETLIST_MINRES_THRESHOLD command.

- If there is only one resistor in the net, it is not merged or shorted.
- If a resistor is attached to a probe node (or *P or *I), that terminal is attached to “keep nodes” and should not be affected.
- The NETLIST_MINRES_HANDLING command does not preserve point-to-point resistance.
- The NETLIST_MINRES_HANDLING command ensures that no resistors exist with their terminals shorted.
- You can specify either SHORT or MERGE in the NETLIST_MINRES_HANDLING command, but not both. If you specify both, the second one overrides the first one.
- The NETLIST_MINRES_HANDLING:MERGE command does not work on resistor nodes with more than 3 branches. In other words, only series merging is supported. However, the command works on all resistor nodes and shorts the nodes appropriately.
- When the NETLIST_MINRES_HANDLING:MERGE command is specified, the capacitance on the reduced node is moved to the smaller resistor.

See Also

- NETLIST_MINRES_THRESHOLD

Feedback

NETLIST_MINRES_THRESHOLD

Merges or shorts all resistances in the netlist less than or equal to the specified threshold. Valid only for transistor-level extraction.

Syntax

```
NETLIST_MINRES_THRESHOLD: threshold_value
```

Arguments

Argument	Description
threshold_value	Threshold at which all resistances are merged or shorted if less than or equal to this value Default: 0

Description

This command merges or shorts all resistances in the netlist less than or equal to the specified threshold.

- You cannot specify a value less than zero.
- This option is governed by the `NETLIST_MINRES_HANDLING: SHORT | MERGE` command.

Examples

```
NETLIST_MINRES_THRESHOLD: 0.001
```

See Also

- NETLIST_MINRES_HANDLING

Feedback

NETLIST_MOVE_SPICE_TYPE_TO_LAST

Specifies how to include the SPICE card character in the output.

Syntax

```
NETLIST_MOVE_SPICE_TYPE_TO_LAST: YES | NO
```

Arguments

Argument	Description
YES	Writes the SPICE type at the lowest level of hierarchy
NO (default)	Writes the SPICE type at the highest level of hierarchy

Description

By default, the SPICE type is added as the first character of the instance name at the highest level of hierarchy in the instance section of an extracted netlist.

Set this command to YES to write the SPICE type only in the instance name at the lowest level of hierarchy.

Examples

For example, a design that contains multiple levels of hierarchy might include lines in the netlist as follows:

```
XXIO.MM0
XXIO.MM0_NETTRAN_2
```

If you set the NETLIST_MOVE_SPICE_TYPE_TO_LAST command to YES, the lines are written as follows:

```
XIO.XMM0
XIO.XMM0_NETTRAN_2
```

Feedback

NETLIST_NAME_MAP

Controls name mapping for SPEF netlists.

Syntax

```
NETLIST_NAME_MAP: YES | NO
```

Arguments

Argument	Description
`YES` (default)	Enables name mapping in SPEF netlists
`NO`	Disables name mapping in SPEF netlists

Description

The `NETLIST_NAME_MAP` command enables or disables name mapping in SPEF netlists. Name mapping reduces the netlist size for big designs by replacing long strings with short codes to indicate nets.

If name mapping is enabled, the netlist file contains a section that lists the name codes, as follows:

```
*NAME_MAP

*3 VDD
*8 X1
*9 X2
...
```

Disabling name mapping greatly increases the netlist size.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_NODE_SECTION

Generates the *|S or *N statements in the output netlist.

Syntax

```
NETLIST_NODE_SECTION: YES | NO
```

Arguments

Argument	Description
YES	Generates *|S or *N statements in the output netlist
NO (default)	Does not generate *|S or *N statements in the output netlist

Description

This command generates the *|S or *N statements in the output netlist.

Using the default setting of NO greatly reduces the netlist size and is useful for most postextraction flows.

Specify this command with netlist formats SPF or SPEF.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_NODENAME_NETNAME

Retains a net name for one of the subnodes of a nonport net. Valid only for transistor-level extraction.

Syntax

```
NETLIST_NODENAME_NETNAME: YES | NO
```

Arguments

Argument	Description
YES	Retains net names for one of the subnodes of a nonport net. Names the subnodes with the net name without the pin delimiter and positive integer.
NO (default)	Does not retain subnode names

Description

The NETLIST_NODENAME_NETNAME command retains a net name for one of the subnodes of a nonport net. The StarRC tool chooses one subnode from a nonport (internal) net and converts it to a net name. This facilitates probing of nonport (internal) nets during simulation.

This command is valid for all settings of the NETLIST_FORMAT command and is particularly useful for the standard parasitic format. However, parasitic netlist reading tools that adhere strictly to the SPEF standard might issue errors. To avoid SPEF netlist reading errors, set the NETLIST_NODENAME_NETNAME command to NO.

Note:

Do not use a probe name specified in a probe text file that is the same as a net name. In that case, the two nodes are electrically shorted.

If a net has a top-level port node, for example, *|P (DSPF) or *P (SPEF), the NETLIST_NODENAME_NETNAME:YES command does not rename or generate a node for that net.

When a net has at least one *|S node (DSPF) or *N node (SPEF), one of those *|S or *N is renamed to match the *|NET or *D_NET net name.

To apply this operation to dangling ports, you must set the NETLIST_NODENAME_NETNAME command to YES and also specify the NODENAME_NETNAME_ON_DANGLING_PORTS command.

Feedback

A node is never a candidate for renaming if it is from one of the following categories:

Node	
Instance port	*|I (DSPF) or *I (SPEF)
Top-level port	*|P (DSPF) or *P (SPEF)
Observation port	**|O (DSPF) or **O (SPEF)
Probe text	**|OP (DSPF) or **OP (SPEF)

If a net contains no *|S or *N subnodes but has at least two *|I nodes, a new node is generated whose name matches the net name.

If a net is floating (no *|P or *|I nodes) or dangling (only one *|P or *|I node), no resistor is generated and no node is renamed.

Examples

```
   NETLIST_NODENAME_NETNAME: NO
*|NET x0.x38.n15 0.000900241PF
*|I (x0.x38.M2|DRN x0.x38.M2 DRN B 0 -492.5 11) //
$llx=-492.5 $lly=4.5 $urx=-489.5 $ury=17.5 $lvl=7
*|I (x0.x38.M1|DRN x0.x38.M1 DRN B 0 -489.5 11)//
$llx=-489.5 $lly=11 $urx=-489.5 $ury=11 $lvl=7
Cg1 x0.x38.M2|DRN 0 2.85306e-16
R1 x0.x38.M2|DRN x0.x38.M1|DRN 0.001  $l=3 $w=10 $lvl=7

Example continues …
       NETLIST_NODENAME_NETNAME: YES
*|NET x0.x38.n15 0.000900241PF
*|I (x0.x38.M2|DRN x0.x38.M2 DRN B 0 -492.5 11) //
$llx=-492.5 $lly=4.5 $urx=-489.5 $ury=17.5 $lvl=7
*|I (x0.x38.M1|DRN x0.x38.M1 DRN B 0 -489.5 11) //
$llx=-489.5 $lly=11 $urx=-489.5 $ury=11 $lvl=7
*|S (x0.x38.n15 -492.5 11//
$llx=-492.5 $lly=4.5 $urx=-489.5 $ury=17.5 $lvl=7)
Cg1 x0.x38.M2|DRN 0 2.85306e-16
R1 x0.x38.n15 x0.x38.M1|DRN 0.001 $l=3 $w=10 $lvl=7
R2 x0.x38.n15 x0.x38.M2|DRN 0.001 $l=3 $w=10 $lvl=7
```

See Also

- NETLIST_FORMAT
- NODENAME_NETNAME_ON_DANGLING_PORTS
- NETLIST_SORT_PIN_NODE
- NETLIST_SUBNODE_SELECTION

Feedback

NETLIST_OUTPUT_DRIVERS

Identifies the driving cell models in the SPEF netlist format for both output and bidirectional instance pins with the optional *D statement.

Syntax

```
NETLIST_OUTPUT_DRIVERS: YES | NO
```

Arguments

Argument	Description
`YES` (default)	Prints the driving cell model name of both output and bidirectional instance pins in the netlist
`NO`	Does not enable the command function

Description

A pin (*I) can be input, output, or bidirectional. Many static timing tools do not require this information for the output and bidirectional pins, because the loading cap for the net is provided. The StarRC tool prints the *D statements for the output and bidirectional instance pins by default.

This command has an effect only if you specify the `NETLIST_FORMAT: SPEF` command.

See Also

- NETLIST_FORMAT
- NETLIST_INPUT_DRIVERS

Feedback

NETLIST_PARASITIC_RESISTOR_MODEL

Writes parasitic resistor model names into the parasitic netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_PARASITIC_RESISTOR_MODEL: YES |NO
```

Arguments

Argument	Description
YES	Writes model information to the parasitic netlist
NO (default)	Does not write layer or model information to the netlist

Description

This command writes resistor models into the parasitic netlist for use by simulators.

The model name is based on the database layer from which the resistor was extracted. If you want to use the same model for more than one nxtgrd file layer, map the corresponding database layers to the same model in the mapping file.

If a nonphysical resistor is introduced into the netlist, it is not generated in the netlist.

If the NETLIST_PARASITIC_RESISTOR_MODEL command is set to YES and you do not specify a resistor model in the mapping file for a layer, no model name is written into the netlist for resistors in that layer and the StarRC tool issues warning messages.

If the REDUCTION command is set either to LAYER_NO_EXTRA_LOOPS, LAYER, or NO and the NETLIST_PARASITIC_RESISTOR_MODEL command is set to YES, the tool writes model names in the netlist for resistors.

Examples

The mapping file uses the following syntax:

```
Conducting Layers
   database_layer GRD_layer RPSQ = value MODEL = model_name
   database_layer GRD_layer MODEL = model_name
   database_layer GRD_layer MODEL = model_name RPSQ = value
Via_layers
   database_layer GRD_layer RPV=value AREA=value MODEL=model_name
   database_layer GRD_layer MODEL=model_name
   database_layer GRD_layer MODEL=model_name RPV=value
AREA=value
```

Example 1

```
TCAD_GRD_FILE: process.nxtgrd
NETLIST_PARASITIC_RESISTOR_MODEL: YES
NETLIST_TAIL_COMMENTS: YES

conducting_layers
     metal2          m2       rpsq=0.02          model=M2R
     metal1          m1       rpsq=0.02          model=M1R
     poly           PO1       rpsq=10            model=PR
     pgate          PO1       rpsq=10            model=PR
     ngate          PO1       rpsq=10            model=PR
```

Example 1 Final Netlist

```
*|NET IN2 0.0253958PF
*|I (xSD_11/M1:GATE xSD_11/M1 GATE I 2e-14 -5.1 29.8)
*|P (IN2 B 0 -30.8 7)
*|I (xSD_11/M6:GATE xSD_11/M6 GATE I 2e-14 -5.1 -7.2)
*|I (xSD_11/M2:GATE xSD_11/M2 GATE I 2e-14 -12.1 29.8)
*|I (xSD_11/M5:GATE xSD_11/M5 GATE I 2e-14 -12.1 -7.2)
Cg3 xSD_11/M1:GATE 0 1.0243e-15
C4 IN2:1 xSD_11/SN_22:1 6.78185e-17
Cg5 IN2:1 0 6.15523e-15
Cg6 IN2 0 3.06582e-15
C7 xSD_11/M6:GATE xSD_11/SN_22:1 2.54972e-16
Cg8 xSD_11/M6:GATE 0 3.78517e-16
Cg9 xSD_11/M2:GATE 0 8.35609e-16
C10 xSD_11/M5:GATE xSD_11/M6:DRN 2.46994e-16
Cg11 xSD_11/M5:GATE 0 6.85312e-16
Cg12 IN2:3 0 1.05002e-14
R3 xSD_11/M1:GATE IN2:1 PR r=130.419 $l=21.8 $w=1 $lvl=3
R4 IN2:1 IN2:2 M2R r=0.0701772 $l=7 $w=2.56 $lvl=1
R5 IN2:1 xSD_11/M6:GATE M1R r=121.333 $l=15.2 $w=1 $lvl=2
R6 IN2 IN2:2 M2R r=0.102702 $l=15.2 $w=3.6 $lvl=1
R7 IN2:2 IN2:3 VIAR r=0.0237957 $a=2.56 $lvl=10
R8 xSD_11/M2:GATE IN2:3 PR r=130.419 $l=21.8 $w=1 $lvl=3
R9 xSD_11/M5:GATE IN2:3 PR r=121.333 $l=15.2 $w=1 $lvl=3
```

See Also

- NETLIST_TAIL_COMMENTS
- REDUCTION

Feedback

NETLIST_PASSIVE_PARAMS

Specifies the generation of passive device parameters in the parasitic or ideal netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_PASSIVE_PARAMS: YES | NO
```

Arguments

Argument	Description
YES	Generates the passive device parameters in the netlist
NO (default)	Does not generate the passive device parameters in the netlist

Description

Selects a format for the designed R, L, or C devices in the parasitic netlist instance section (NETLIST_FILE) and the ideal netlist (NETLIST_IDEAL_SPICE_FILE).

The following is an example of the default format for an RLC instance netlist. This format does not require a SPICE model for these devices for simulation:

```
R11 R11:A R11:B 1000 $.model=Rdev $BULK=VDD
C12 C12:A C12:B 1e-12 $.model=Cdev
L13 L13:A L14:A 10 $.model=Ldev
```

If the NETLIST_PASSIVE_PARAMS:YES command is set, the format changes to include any properties you calculated in the Hercules or IC Validator runset as well as the standard device extraction properties always calculated by Hercules or IC Validator.

Nonstandard properties such as capacitor length and width are always generated as comments in the netlist, because they are not SPICE-model-compatible. Similarly, the BULK terminals on ideal RLC devices are always generated as comments in the netlist.

Examples

The following is an example of the NETLIST_PASSIVE_PARAMS: YES format:

```
R11 R11:A R11:B Rdev r=1000 l=10u w=1000um
   $BULK=VDD
C12 C12:A C12:B Cdev c=1e-12 area=100p pj=40u
   $l=10u $w=10u
L13 L13:A L13:B Ldev l=10
```

Feedback

See Also

- NETLIST_FILE
- NETLIST_INSTANCE_SECTION
- NETLIST_IDEAL_SPICE_FILE

Feedback

NETLIST_PIC_LEVEL_MAP

Reports the PIC level of a map in a netlist.

Syntax

```
NETLIST_PIC_LEVEL_MAP: YES | NO
```

Arguments

Argument	Description
YES	Reports the PIC level of a map in the netlist
NO (default)	Does not report the PIC level of a map in the netlist

Feedback

NETLIST_POSTPROCESS_COMMAND

Specifies a method to modify an ASCII output parasitic netlist.

Syntax

```
NETLIST_POSTPROCESS_COMMAND: script_file
```

Arguments

Argument	Description
script_file	Script file to be run on the netlist

Description

The NETLIST_POSTPROCESS_COMMAND command gives you the flexibility to modify an ASCII netlist by operating on it with a script. Operations take place in real time. It is not possible to perform any operations on the entire netlist after it is saved. OA netlists cannot be postprocessed; in this case, the StarRC tool ignores the command and issues a warning message.

The argument must be an executable, either an utility program or a script. If the specified executable is not in a directory specified by the PATH environment variable, you must specify the full path to the executable. The StarRC tool does not check for the existence of the executable. If the specified file is not found, the run exits with an error message.

This command can be used with the NETLIST_COMPRESS_COMMAND command to perform two operations on a netlist. Only one instance of each command is allowed in the command file. If both commands are present, the NETLIST_POSTPROCESS_COMMAND operation is performed first regardless of the command order.

See Also

- NETLIST_COMPRESS_COMMAND

Feedback

NETLIST_POWER_FILE

Controls the file name given to the file created by `POWER_EXTRACT:RONLY`, which contains the power rail resistor values.

Syntax

```
NETLIST_POWER_FILE: file_name
```

Arguments

Argument	Description
file_name	The file name of a netlist (for power) with resistors only Default: none

Description

This command should be used only with the `POWER_EXTRACT:RONLY` command.

See Also

- POWER_EXTRACT

Feedback

NETLIST_PRECISION

Specifies the number of significant digits to report for parasitic element values.

Syntax

```
NETLIST_PRECISION: sig_digits
```

Arguments

Argument	Description
sig_digits	The number of significant digits to report Default: 6

Description

The NETLIST_PRECISION command sets the number of significant digits used for reporting parasitics values in output netlists and in files generated by the PLACEMENT_INFO_FILE command. Trailing zeros are not displayed.

The NETLIST_PRECISION command does not affect the reporting of xy coordinates and device dimensions. The number of digits after the decimal point for geometric parameters (such as the x- and y-coordinates of nodes and resistors, the length and width of resistors, and the area of vias) is based on the precision of the input design files, as follows:

- For design precision of 1 nm, the number of digits after the decimal point is 3.
- For design precision of 0.1 nm or smaller, the number of digits after the decimal point is 4.

Examples

The following netlist excerpt shows output with the default of 6:

```
*|NET x0/n36 0.0115668PF
*|I (x0/x37:A x0/x37 A B 0 -479 23.5)
Cg3 x0/n36:1 0 2.60786e-15
R5 x0/n36:1 x0/n36:4 0.001
```

The NETLIST_PRECISION: 10 command results in the following output:

```
*|NET x0/n36 0.01156682055PF
*|I (x0/x37:A x0/x37 A B 0 -479 23.5)
Cg3 x0/n36:1 0 2.607864308e-15
R5 x0/n36:1 x0/n36:4 0.001000000047
```

Feedback

See Also

- PLACEMENT_INFO_FILE

Feedback

NETLIST_PRINT_CC_TWICE

Specifies whether to generate the reciprocal coupling capacitor twice in the netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_PRINT_CC_TWICE: YES | NO
```

Arguments

Argument	Description
YES	Generates the reciprocal coupling capacitor twice in the netlist
NO (default)	Does not generate the reciprocal coupling capacitor twice in the netlist

Description

The second capacitor is reported as a comment so that the netlist remains accurate for standard simulation and timing analysis. This command is used when the netlist format is specified as STAR or NETNAME.

Examples

```
NETLIST_PRINT_CC_TWICE: NO
   *|NET NETA 0.0010000PF
   *|I (NETA:F1 I0 A I 0 485.5 11)
   *|I (NETA:F2 I1 Z O 0 483.5 11)
   *|I (NETA:F2 I1 Z O 0 483.5 11)
   R1 NETA:F1 NETA:F2 12.43
   C1 NETA:F1 0 6e-15
   C2 NETA:F2 0 3.5e-15
   C3 NETA:F1 NETB:F1 5e-16
   *|NET NETB 0.007000PF
   *|P (NETB B 0 32.5 8.3)
   *|I (NETB:F1 I32 B I 0 554.3 12)
   RNETB NETB:F1 1032
   C4 NETB 0 5e-15
   C5 NETB:F1 0 1.5e-15
   NETLIST_PRINT_CC_TWICE: YES
   *|NET NETA 0.0010000PF
   *|I (NETA:F1 I0 A I 0 485.5 11)
   *|I (NETA:F2 I1 Z O 0 483.5 11)
   R1 NETA:F1 NETA:F2 12.43
   C1 NETA:F1 0 6e-15
   C2 NETA:F2 0 3.5e-15
   C3 NETA:F1 NETB:F1 5e-16
   *|NET NETB 0.007000PF
```

```
*|P (NETB B 0 32.5 8.3)
*|I (NETB:F1 I32 B I 0 554.3 12)
RNETB NETB:F1 1032
C4 NETB 0 5e-15
C5 NETB:F1 0 1.5e-15
*C6 NETB:F1 NETA:F1 5e-16
```

See Also

- COUPLE_TO_GROUND
- NETLIST_FORMAT

Feedback

NETLIST_REMOVE_DANGLING_BRANCHES

Removes dangling RC branches in the netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_REMOVE_DANGLING_BRANCHES: YES | NO
```

Arguments

Argument	Description
YES	Removes dangling branches
NO (default)	Does not remove dangling branches

Description

The NETLIST_REMOVE_DANGLING_BRANCHES command removes dangling RC branches in the netlist. Capacitors on dangling branches—both grounded and coupling capacitors—are moved to an appropriate location in the RC network.

The NETLIST_REMOVE_DANGLING_BRANCHES command

- Cannot be used for SPEF netlists
- Does not affect point-to-point resistance between *P or *I elements, because the removed RC branch is dangling
- Does not work on open nets that have multiple RCGs

Note:

The likelihood of having a dangling RC branch when the REDUCTION command is set to YES, HIGH, or NO_EXTRA_LOOPS is low, because the reduction operation eliminates dangling branches.

See Also

- NETLIST_MINRES_HANDLING
- NETLIST_MINRES_THRESHOLD
- REDUCTION

Feedback

NETLIST_RENAME_PORTS

Specifies the global renaming scheme for ports.

Syntax

```
NETLIST_RENAME_PORTS: XY | delimiter
```

Arguments

Argument	Description
XY	Adds a suffix based on the instance port cell location
XYI	Adds a suffix based on the instance port cell location and a randomly generated short integer
delimiter (default)	Specifies a delimiter string to use for naming Default: _ (single underscore character)

Description

The NETLIST_RENAME_PORTS command specifies the global renaming scheme for multiple port instances, which might exist when any of the following commands are used:

- SHORT_PINS: NO
- INSTANCE_PORT: NOT_CONDUCTIVE
- INSTANCE_PORT: CONDUCTIVE SUFFIXED MULTIPLE

The default scheme to name multiple ports is a single underscore character followed by sequential numbering. You can use the NETLIST_RENAME_PORTS command to replace the default delimiter with a different character or string. The XY mode names each port with the cell local coordinates of the instance port interaction point, in nanometers. The XYI mode also uses the coordinates, but adds a short integer to avoid name duplication when the port has an identical location on different layers.

When you use the following modes with the NETLIST_RENAME_PORTS command along with the port marker shapes in the layout:

- *delimeter*: Uses the center of the port marker shape as the port location.
- XY: Uses the lower-left coordinates as the port location.

Feedback

Examples

The examples in this section use the same test case with different invocations of the NETLIST_RENAME_PORTS command. In all cases, the following commands are also used in the command file:

```
NETLIST_FORMAT: SPEF
INSTANCE_PORT: CONDUCTIVE MULTIPLE SUFFIXED
NETLIST_NAME_MAP: NO
```

The INSTANCE_PORT command preserves multiple instances for the same port. The NETLIST_NAME_MAP:NO command is used for clarity in these examples but does not affect the operation of the NETLIST_RENAME_PORTS command.

NETLIST_RENAME_PORTS: (with no argument or not used at all)

```
*PORTS

VDD O

//*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_1 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800 ...
*I X1:VDD_2 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD_1 B *C 29.5200 13.9400 *D INV_Q// $llx=29.5200 ...
```

NETLIST_RENAME_PORTS:#

```
*PORTS

VDD O

//*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD#1 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800 ...
*I X1:VDD#2 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD#1 B *C 29.5200 13.9400 *D INV_Q// $llx=29.5200 ...
```

NETLIST_RENAME_PORTS:_snpsindex_

```
*PORTS

VDD O

//*LAYER_MAP
```

Feedback

```
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_snpsindex_1 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800 ...
*I X1:VDD_snpsindex_2 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD_snpsindex_1 B *C 29.5200 13.9400 *D INV_Q// $llx=29.5200 ...
```

NETLIST_RENAME_PORTS: XY

```
*PORTS

VDD_x200y13910 O

//*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_x10000y10500 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800 ...
*I X1:VDD_x0y10500 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD_x200y13910 O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD_x0y10500 B *C 29.5200 13.9400 *D INV_Q// $llx=29.5200 ...
```

NETLIST_RENAME_PORTS: XYI

```
*PORTS

VDD_x200y13910_7302 O

//*LAYER_MAP
...
*D_NET VDD 1.67634

*CONN
*I X1:VDD_x10000y10500_46410 B *C 19.1800 13.9400 *D INV_Q// $llx=19.1800
*I X1:VDD_x0y10500_55973 B *C 9.18000 13.9400 *D INV_Q// $llx=9.18000 ...
*P VDD_x200y13910_7302 O *C 0.200000 13.9100 // $llx=13.4400 ...
*I X2:VDD_x0y10500_5375 B *C 29.5200 13.9400 *D INV_Q// $llx=29.5200 ...
```

See Also

- INSTANCE_PORT
- SHORT_PINS
- NETLIST_NAME_MAP

Feedback

NETLIST_RESISTANCE_UNIT

Specifies the units used for reporting resistance values in both the header and the body of the output netlist.

Syntax

```
NETLIST_RESISTANCE_UNIT: res_unit
```

Arguments

Argument	Description
res_unit	Resistance unit in a SPEF netlist Units: ohm Default: 1.0

Description

This command specifies the units used for reporting resistance values in both the header and the body of SPEF netlists.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_SELECT_NETS

Specifies a subset of extracted nets to report in the output netlist.

Syntax

```
NETLIST_SELECT_NETS: net_names
```

Arguments

Argument	Description
net_names	List of extracted nets to include in the parasitic netlist Default: all nets (*)

Description

This command specifies a subset of the extracted nets to include in the output netlist. Wildcards “*”, “!”, and “?” are supported. The same selection rules detailed in the NETS command reference page also apply to this command.

The StarRC tool issues a warning message if a net marked with the NETLIST_SELECT_NETS command was not extracted because it was not specified by the NETS command or because it does not exist.

See Also

- NETLIST_TYPE
- NETS

Feedback

NETLIST_SIM_OPTIONS

Specifies text to add to the output netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_SIM_OPTIONS: text_line
```

Arguments

Argument	Description
text_line	A line of text to be written into the output netlist

Description

This command allows you to specify lines of text to be written directly into the output netlist. Each occurrence of the NETLIST_SIM_OPTIONS command writes a line of text, in the order specified. Typical usage of this feature is to set simulation options to be interpreted by downstream analysis tools.

Examples

The following commands add custom lines to the output netlist:

```
NETLIST_SIM_OPTIONS: .param cflag=0
NETLIST_SIM_OPTIONS: .param rflag=0
NETLIST_SIM_OPTIONS: .option scale=0.9
```

The output netlist might look something like the following example. The custom lines appear after the file header and comments inserted by the NETLIST_COMMENTS_FILE command:

```
** TCAD_GRD_FILE process.nxtgrd
**   TCAD_TIME_STAMP Tue Nov 27 15:29:49 2007
**   TCADGRD_VERSION 62
…
.param cflag=0
.param rflag=0
.option scale=0.9
*|GROUND_NET 0
*|NET BOUNDBUF_N_838 0.0735652PF
*|I (cg/p0849A134:Z cg/p0849A134 Z O 0 -314.774 -248.309)
…
```

Feedback

See Also

- NETLIST_FORMAT
- NETLIST_COMMENTS_FILE

Feedback

NETLIST_SMC_FORMULA

This command specifies the use of formulas for writing RC values in the output netlist in a simultaneous multicorner flow. Valid only for transistor-level extraction.

Syntax

```
NETLIST_SMC_FORMULA: YES | NO
```

Arguments

Argument	Description
YES	A single final netlist is generated that contains RC values written as formulas using the corner names as variables
NO (default)	One or more final netlist files are generated as defined by the SELECTED_CORNERS command

Description

In a simultaneous multicorner flow, the NETLIST_SMC_FORMULA command specifies that the output should be a single netlist containing RC values written as formulas that use the corner names as variables. Any simulation tool that reads the netlist file can set the corner name variables to 1 or 0 to calculate the RC values for a specific corner. The allowable netlist formats for this option are SPF, NETNAME, and OA.

Examples

When the NETLIST_SMC_FORMULA: YES command is used in a flow in which three corners are defined (NOM_T1, NOM_T2, and RCMAX_T3), the SPF file output contains lines similar to the following example:

```
*|NET A '0.63*NOM_T1+0.65*NOM_T2+0.75*RCMAX_T3'PF
Cg1 A:1 0 3.6e-17*NOM_T1+4.07e-17*NOM_T2+4.09e-17*RCMAX_T3
R162 A:1 A:2 4.8*NOM_T1+4.9*NOM_T2+4.8*RCMAX_T3
```

See Also

- SIMULTANEOUS_MULTI_CORNER
- SELECTED_CORNERS
- NETLIST_FORMAT

Feedback

NETLIST_SORT_PIN_NODE

Sorts only nonport nets with *|I nodes only. Valid only for transistor-level extraction.

Syntax

```
NETLIST_SORT_PIN_NODE: YES | NO
```

Arguments

Argument	Description
YES	Sorts nets with the *|I node only.
NO (default)	Does not sort nets

Description

The NETLIST_SORT_PIN_NODE command sorts, by net name, only those nets with the *|I node. The command does not sort nets with the *|P node.

See Also

- NETLIST_NODENAME_NETNAME
- NETLIST_SUBNODE_SELECTION

Feedback

NETLIST_SUBCKT

Specifies whether to write the .SUBCKT and .ENDS statements in an SPF netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_SUBCKT: YES | NO
```

Arguments

Argument	Description
`YES` (default)	Writes the .SUBCKT and .ENDS statements
`NO`	Does not write the .SUBCKT and .ENDS statements

Description

Specifies whether to write the .SUBCKT and .ENDS statements in standard parasitic format (SPF) netlists.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_SUBNODE_SELECTION

Sorts nonport nets to select a nonport net with the smallest coordinate. Valid only for transistor-level extraction.

Syntax

```
NETLIST_SUBNODE_SELECTION: YES | NO | MOSGATE
```

Arguments

Argument	Description
`MOSGATE`	Sorts nonport nodes of which the subnode of resistor is connected to the non-finger gate.
`YES`	Sorts nonport nets to select the nonport net with the smallest coordinate. Adds net-name node to the selected nonport net, and appends the selected nonport net with the 0.001 ohm resistor integer.
`NO` (default)	Does not sort subnode names.

Description

To retain the net name of the subnode of the nonport (internal) net with the smallest coordinate, the `NETLIST_SUBNODE_SELECTION: YES | MOSGATE` command

1. Sorts all nonport nets in a design
2. Selects the nonport net with the smallest coordinate
3. Adds the net-name node to the selected nonport net by using the subnode of the selected nonport net

   This helps to probe the nonport nets during simulation.
4. Appends the selected nonport net with the 0.001 ohm resistor integer

When you use the `NETLIST_SUBNODE_SELECTION: YES` command to specify a net without the *|P node, the StarRC tool

1. Sorts the nonport nets with the *|I node by coordinates
2. Generates a subnode by determining the coordinate of the first *|I node on the net

Feedback

When you use the NETLIST_SUBNODE_SELECTION: MOSGATE command to specify a net without the *|P node, the StarRC tool

1. Sorts the nonport nets with the *|I node by coordinates
2. Identifies the first non-finger gate of all the gates
3. Searches the first resistor connected to the gate
4. Renames the node on the other side of the resistor according to a net name

This command is valid for all settings of the NETLIST_FORMAT command and is particularly useful for the standard parasitic format. However, parasitic netlist reading tools that adhere strictly to the SPEF standard might issue errors. To avoid SPEF netlist reading errors, set the NETLIST_SUBNODE_SELECTION command to NO.

Note:

Do not use a probe name specified in a probe text file that is the same as a net name. In that case, the two nodes are electrically shorted.

If a net has a top-level port node, for example, *|P (DSPF) or *P (SPEF), the NETLIST_SUBNODE_SELECTION:YES command does not rename or generate a node for that net.

When a net has at least one *|S node (DSPF) or *N node (SPEF), one of the *|S or *N node is renamed to match the *|NET or *D_NET net name.

To apply this operation to dangling ports, you must set the NETLIST_SUBNODE_SELECTION command to YES and also specify the NODENAME_NETNAME_ON_DANGLING_PORTS command.

A node is never a candidate for renaming if it is from one of the following categories:

Node	
Instance port	*|I (DSPF) or *I (SPEF)
Top-level port	*|P (DSPF) or *P (SPEF)
Observation port	**|O (DSPF) or **O (SPEF)
Probe text	**|OP (DSPF) or **OP (SPEF)

If a net contains no *|S or *N subnodes but has at least two *|I nodes, a new node is generated whose name matches the net name.

If a net is floating (no *|P or *|I nodes) or dangling (only one *|P or *|I node), no resistor is generated and no node is renamed.

Feedback

Examples

The following examples shows how the tool sorts nonport nets with the `NETLIST_SUBNODE_SELECTION: YES` command:

```
NETLIST_SUBNODE_SELECTION: YES
   *|NET LN_135 0.0603659PF
   *|I (LD_5/M5:SRC LD_5/M5 SRC B 0 81.0000 355.0000)
   *|I (LD_5/M1:SRC LD_5/M1 SRC B 0 81.0000 392.0000)
   *|I (LD_43/M2:GATE LD_43/M2 GATE I 2e-14 129.5000 547.0000)
   *|I (LD_43/M1:GATE LD_43/M1 GATE I 2e-14 129.5000 584.0000)
   R19_137 LN_135 LD_5/M5:SRC 0.001
   R19_138 LD_5/M5:SRC LN_135:2 0.25

   *|NET LN_74 0.0409189PF
   *|I (LD_34/M5:GATE LD_34/M5 GATE I 2e-14 60.5000 157.0000)
   *|I (LD_34/M1:GATE LD_34/M1 GATE I 2e-14 60.5000 194.0000)
   *|I (LD_8/M4:SRC LD_8/M4 SRC B 0 65.0000 258.0000)
   *|I (LD_8/M10:SRC LD_8/M10 SRC B 0 64.0000 295.0000)
   R44_147 LN_74 LD_34/M5:GATE 0.001
```

The following examples shows how the tool sorts nets with *|I node with the `NETLIST_SUBNODE_SELECTION: MOSGATE` command:

```
NETLIST_SUBNODE_SELECTION: MOSGATE
*|NET L2 0.00939475PF
*|I (LN3/M0@4:GATE LN3/M0@4 GATE I 4.36356e-16 -1723.6945 -2189.0340)
*|I (LN2/M1@4:GATE LN2/M1@4 GATE I 4.36356e-16 -1715.9005 -2189.0340)
*|I (LN4/M1@4:XYZ LN4/M1@4 XYZ B 0 -1773.7755 -2189.0000)
*|I (LN4/M0:XYZ LN4/M0 XYZ B 0 -1773.7755 -2188.8980)
*|I (LN3/M1@3:GATE LN3/M1@3 GATE I 4.36356e-16 -1723.6945 -2188.8640)
*|I (LN2/M0:GATE LN2/M0 GATE I 4.36356e-16 -1715.9005 -2188.8640)
*|I (LN3/M0@3:GATE LN3/M0@3 GATE I 4.36356e-16 -1723.6945 -2188.6940)
*|I (LN2/M1:GATE LN2/M1 GATE I 4.36356e-16 -1715.9005 -2188.6940)
*|I (LN4/M1:XYZ LN4/M1 XYZ B 0 -1773.7755 -2188.6600)
…
*|S (L2:54 -1719.3680 -2187.6280)
*|S (L2:55 -1719.1590 -2189.0340)
*|S (L2:56 -1719.3620 -2189.0340)
*|S (L2 -1719.1590 -2188.8640)
*|S (L2:58 -1719.3620 -2188.8640)
*|S (L2:59 -1719.1590 -2188.6940)
…
R6_1 LN2/M1@4:GATE L2:55 152.588
R6_2 LN2/M1@4:GATE L2:71 152.588
R6_3 LN2/M0:GATE L2 152.588
R6_4 LN2/M0:GATE L2:72 152.588
R6_5 LN2/M1:GATE L2:59 152.588
```

Feedback

See Also

- NETLIST_FORMAT
- NODENAME_NETNAME_ON_DANGLING_PORTS
- NETLIST_NODENAME_NETNAME
- NETLIST_SORT_PIN_NODE

Feedback

NETLIST_SWAP_TERMINAL

Specifies to swap the order of device pins in the output netlist.

Syntax

```
NETLIST_SWAP_TERMINAL: [PORTxy] model_names
```

Arguments

Argument	Description
xy	Two device terminals to swap Values: X and Y must each be an integer from 1 to 9 inclusive Default: 12 (swap terminals 1 and 2)
model_names	Layout model names for which to perform the terminal swap Default: none

Description

The `NETLIST_SWAP_TERMINAL` command specifies two device terminals to swap in the output netlist. If multiple instances of this command overlap, the last command takes precedence.

This command can be useful for downstream simulation tools when the pin order is different between the StarRC tool and the SPICE model.

The following command specifies to swap terminals 2 and 4 in models abc and def:

```
NETLIST_SWAP_TERMINAL: PORT24 abc def
```

If you omit the port specification, the tool swaps terminals 1 and 2. The following command specifies to swap terminals 1 and 2 in model abc:

```
NETLIST_SWAP_TERMINAL: abc
```

Feedback

NETLIST_TAIL_COMMENTS

Writes geometric information about parasitic resistors as comments in the netlist.

Syntax

```
NETLIST_TAIL_COMMENTS: YES | NO
```

Arguments

Argument	Description
YES	Writes resistor geometric information as comments in the netlist
NO (default)	Disables resistor tail comments in the netlist

Description

The NETLIST_TAIL_COMMENTS command controls whether geometric information about parasitic resistors is added to the netlist output.

When the NETLIST_TAIL_COMMENTS command is set to YES,

- The allowed settings for the REDUCTION command are NO, LAYER, LAYER_NO_EXTRA_LOOPS, and TOPOLOGICAL.
- The allowed settings for the POWER_REDUCTION command are NO or YES only if the POWER_EXTRACT command is set to YES.
- The allowed settings for the POWER_REDUCTION commands are NO, LAYER, and LAYER_NO_EXTRA_LOOPS only if the POWER_EXTRACT command is set to RONLY..

Layer Maps in the Netlist

For device-level extraction, the netlist contains a LAYER_MAP section, which contains both the design database layer names and the ITF layers names. Multiple design layer names can map to the same ITF layer.

An example of a LAYER_MAP section is as follows:

```
*LAYER_MAP

*0 SUBSTRATE
*1 M1   ITF=M1
*2 M2   ITF=METAL_2
*3 M3   ITF=METAL_3
*4 POLY1   ITF=POLY
*5 POLY2   ITF=POLY
...
```

Feedback

Layer names might be generated names. Extra layers are formed when there are database layers at the diffusion level or below that share a contact. For example, if the runset contains the following line, the LAYER_MAP section contains an extra layer called nsd:psd or psd:nsd, which becomes the lower terminal layer of the diffCon via resistors.

```
CONNECT metal1 nsd psd BY diffCon
```

Resistor Information

The SPF and NETNAME netlist formats present geometric information as follows, for the example of a conductor resistor R3 and a via resistor R4:

```
R3 n1:3 n1:43 132.4 $l=12.3 $w=0.45 $lvl=3
R4 n1:3 n1:4 1.2 $a=0.6332 $lvl=14 $vdx=0.05 $vdy=0.05
R5 M2M1_22_24:1 M2M1_22_24 22.37 $l=0.5 $w=0.22 $lvl=4 $m=2
```

The SPEF netlist format presents geometric information as follows:

```
3 n1:3 n1:43 132.4 // $l=12.3 $w=0.45 $lvl=3
4 n1:3 n1:4 1.2 // $a=0.6332 $lvl=14 $vdx=0.05 $vdy=0.05
```

Reported parameters include the following:

- The $lvl parameter is a number that appears in the LAYER_MAP section of the netlist, just after the header. The number is associated with a retained mapping file layer name.
- The $vdx and $vdy parameters are via x and y dimensions before merge.
- The $m parameter is the mask number, if extraction is performed for a multiple mask process. The mask number can come from the design database or from the layer mapping file.
- The $w and $l parameters are the resistor dimensions.

  Under most conditions, the minimum resistor width reported is the WMIN value for that layer. However, if the DETECT_FUSE command is set to YES, the actual width is reported.

  The $w and $l parameters reported in tail comments for MOS transistor source and drain regions are not meaningful because these diffusions are extracted in a complex mesh structure.

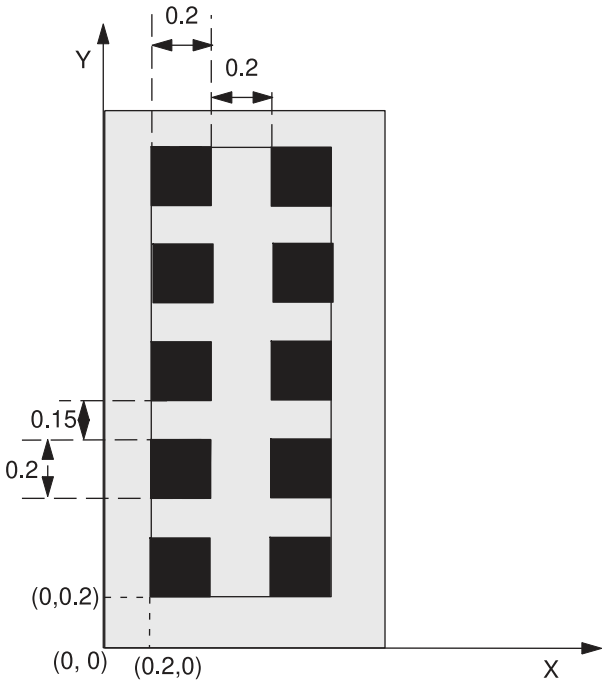
Via Resistor Information

To see the effects of merged vias in an SPF file, you must include the NETLIST_TAIL_COMMENTS: YES, EXTRA_GEOMETRY_INFO: NODE, and KEEP_VIA_NODES: YES commands in the command file.

Figure 215 shows a 5x2 via array. The individual via size is 0.2 microns by 0.2 microns, the vertical via-to-via spacing is 0.15 microns, and the horizontal via-to-via spacing is 0.2

microns. If you specify `MAX_VIA_ARRAY_SPACING=0.3` in the mapping file for the via layer, this via array extracted as one via resistor.

Figure 215 Via Array



If the `NETLIST_TAIL_COMMENTS` command is set to `YES`, an output SPF file contains the following information:

```
R45 29 32 0.2 $a=0.4 $lvl=5 $n=5x2 $p=4.4
```

Nodes 29 and 32 reside on the upper and lower layers connected by the via resistor. The via resistor value is calculated based on the total area of the vias, which is reported as the `$a` parameter. The `$p` parameter is the perimeter of the bounding box of the via array and is calculated as follows:

perimeter = (0.2 x 2 + 0.2) x 2 + (0.2 x 5 + 0.15 x 4) * 2 = 4.4

See Also

- NETLIST_FILE
- NETLIST_FORMAT
- POWER_REDUCTION

Feedback

- POWER_EXTRACT
- REDUCTION
- CAPACITOR_TAIL_COMMENTS

Feedback

NETLIST_TIME_UNIT

Specifies the units for reporting delay values in the SPEF netlist.

Syntax

```
NETLIST_TIME_UNIT: time_unit
```

Arguments

Argument	Description
time_unit	Specifies the unit for delay in the SPEF netlist Units: seconds Default: 1e-09

Description

This command specifies the units for reporting delay values in the header and body of the netlist. You can use this command only with a SPEF netlist.

See Also

- NETLIST_FORMAT

Feedback

NETLIST_TOTALCAP_THRESHOLD

Specifies the capacitance threshold that determines whether a net is reported in the netlist. Valid only for transistor-level extraction.

Syntax

```
NETLIST_TOTALCAP_THRESHOLD: capacitance_value
```

Arguments

Argument	Description
capacitance_value	Threshold below which the net is treated as ideal and there is not any D_NET or *|NET in the parasitic netlist. The capacitance value cannot be less than 0. Units: farads (F) Default: 0.0

Description

This command specifies the capacitance threshold that determines whether a net is reported in the netlist. If the total capacitance is below the specified threshold, then the net is treated as “ideal” and there is no D_NET or *|NET in the parasitic netlist.

Examples

The following example sets the ideal threshold at 0.5 fF:

```
NETLIST_TOTALCAP_THRESHOLD: 0.5e-15
```

Feedback

NETLIST_TYPE

Specifies the parasitics to be written to the netlist for nets identified in the NETLIST_SELECT_NETS command.

Syntax

```
NETLIST_TYPE: [no_couple] [R | CG | CC | RCG | RCC] net_names
```

Arguments

Argument	Description
no_couple	No coupling capacitance is retained from other CC or RCC nets to the listed nets. If this option is not specified, any extracted coupling capacitance is retained from other CC or RCC nets to the listed nets. This option is valid only when used with the R, CG, or RCG options.
R	Resistance only
CG	Lumped capacitance to ground only
CC	Coupled capacitance with lumped capacitance to ground
RCG	Resistance plus lumped capacitance to ground
RCC (default)	Resistance plus coupled capacitance
net_names	Net names to which the specified type applies. Wildcards are allowed.

Description

The last NETLIST_TYPE command specified for a particular net overrides previous NETLIST_TYPE specifications for the same net. When the NETLIST_TYPE command is not specified, parasitics are generated according to the EXTRACTION and COUPLE_TO_GROUND settings.

If the NETLIST_TYPE command is set to CC or RCC, couplings between that net and all other selected nets are included in the netlist regardless of the NETLIST_TYPE setting for those other selected nets, unless no_couple is specified for those other selected nets.

See Also

- NETLIST_COUPLE_UNSELECTED_NETS
- NETLIST_SELECT_NETS

Feedback

NETLIST_UNSCALED_COORDINATES

Specifies whether to use scaled or unscaled coordinates in the netlist and in certain summary reports.

Syntax

```
NETLIST_UNSCALED_COORDINATES: YES | NO
```

Arguments

Argument	Description
YES	Writes unscaled coordinates in the netlist
NO (default)	Writes scaled coordinates in the netlist

Description

For debugging issues such as opens, shorts, and SMIN violations, unscaled (original layout) coordinates are useful. The StarRC command file NETLIST_UNSCALED_COORDINATES and MAGNIFICATION_FACTOR commands and the ITF file HALF_NODE_SCALE_FACTOR command affect coordinate scaling in the opens.sum, shorts_all.sum, smin.sum, and vias.sum files.

Table 81 summarizes the interaction between these commands.

Table 81 Coordinate Scaling in Netlist and Summary Reports

HALF_NODE_ SCALE_FACTOR	MAGNIFICATION_ FACTOR	NETLIST_UNSCALED_ COORDINATES	Coordinates
value	value	ignored	n/a (error)
not set	value	YES	unscaled
not set	value	NO (default)	scaled
value	not set	not set (StarRC sets to YES automatically)	unscaled
value	not set	YES	unscaled
value	not set	NO	scaled

Feedback

See Also

- HALF_NODE_SCALE_FACTOR
- MAGNIFICATION_FACTOR

Feedback

NETLIST_UNSCALED_RES_PROP

Specifies the output of scaled or unscaled resistor properties.

Syntax

```
NETLIST_UNSCALED_RES_PROP: YES | NO
```

Arguments

Argument	Description
YES	Writes unscaled resistor properties in the netlist
NO (default)	Writes scaled resistor properties in the netlist

Description

The NETLIST_RES_PROP command affects the reporting of resistor properties for all parasitic resistors. If you set this command to YES, unscaled values are reported for the following properties: length ($l) and width ($w) for resistors in conductor layers; area ($a) for unmerged via resistors; area ($a), via merge dimensions ($vdx, $vdy), and perimeter ($p) for merged via resistors; and bounding box coordinates ($llx, $lly, $urx, $ury) for all resistors.

Table 82 shows the effect of ITF and StarRC commands on resistor property scaling.

Table 82 Resistor Property Scaling

HALF_NODE_SCALE_FACTOR	MAGNIFICATION_FACTOR	NETLIST_UNSCALED_RES_PROP	Resistor Properties
value	value	ignored	n/a (error)
not set	value	YES	unscaled
not set	value	NO (default)	scaled
value	not set	not set (StarRC sets to YES automatically)	unscaled
value	not set	YES	unscaled
value	not set	NO	scaled

In addition, the NETLIST_TAIL_COMMENTS: YES command is required to include resistor properties in the parasitic netlist and the EXTRA_GEOMETRY_INFO: NODE RES command is required to report node and resistor coordinates.

Feedback

See Also

- HALF_NODE_SCALE_FACTOR
- MAGNIFICATION_FACTOR
- NETLIST_TAIL_COMMENTS
- EXTRA_GEOMETRY_INFO

Feedback

NETLIST_USE_M_FACTOR

Specifies whether to use the magnification factor from the schematic netlist to calculate device properties. Valid only for transistor-level extraction.

Syntax

```
NETLIST_USE_M_FACTOR: YES | NO
```

Arguments

Argument	Description
YES (default)	Calculates device properties with a magnification factor
NO	Does not calculate device properties with a magnification factor

Description

When this command is set to YES, it uses the magnification factor from the schematic netlist to calculate device properties.

Note:

The NETLIST_USE_M_FACTOR command has an effect only when the XREF:COMPLETE command is also specified. The value is ignored for any other setting.

See Also

- XREF

Feedback

NETS

Specifies a list of nets to extract.

Syntax

`NETS: net1 net2 ...`

Arguments

Argument	Description
net1 net2 ...	Nets to be extracted, separated by spaces Default: all nets (*)

Description

By default, the StarRC tool extracts all nets, which is the equivalent of the `NETS:*` command. This is the behavior when the `NETS` command is not used in the StarRC command file.

However, if you use the `NETS` command even one time, the tool extracts only the nets that are specified in a `NETS` command. If the `NETS` command appears multiple times, the list of nets to extract is a cumulative list based on the contents of all of the `NETS` commands.

You can also use the `NETS_FILE` command to specify a file that contains a list of `NETS` commands. The effect of the `NETS` command is the same whether it appears directly in the command file or in a file specified by the `NETS_FILE` command.

The following usage notes apply:

- Wildcards are supported: asterisk (*) for all values, question mark (?) for a single character, and exclamation mark (!) for negation.
- If a `NETS:*` command appears anywhere within the command file or in a file specified with the `NETS_FILE` command, the tool extracts all nets.
- A `NETS` command that does not specify at least one net for extraction returns an error for gate-level extraction. An example is the !* argument when used alone. However, if the same `NETS` command specifies at least one net for extraction, that net specification overrides the !* argument.
- Net names that originate from a hierarchical netlist must be fully flattened with the hierarchical separator defined by the `HIERARCHICAL_SEPARATOR` command. Additionally, any reserved character from the input database must be included in this list to allow the use of special characters such as the `BUS_BIT` delimiter.
- Names must be case-sensitive in accordance with the `CASE_SENSITIVE` command.

The StarRC tool does not alter the net information in the input database except to flatten names as required. The case of names in the input database is always preserved in the output netlist. It is important to understand how the place and route tool handles names. If possible, extract names directly from the input database.

Table 83 describes the behavior of the NETS command for different combinations of input that include wildcards.

Table 83 Behavior of Wildcards in the NETS Command

Argument	NETS command behavior
*	Selects all nets (default)
* A B	Selects all nets
* !A	Selects all nets except net A
* !XY*	Selects all nets except nets with names beginning with XY
!* or !* !A or !A	Gate-level extraction: Returns an error , because there must be at least one net for extraction Transistor-level extraction: Generates output with no parasitics
!* A	Selects only net A
!* XY*	Selects only nets with names beginning with XY

For transistor-level extraction, the behavior of the NETS command when the argument list does not select any nets (including settings such as !*, !* !A, and !A) depends on the setting of the NETLIST_INSTANCE_SECTION command, as follows:

- If the NETLIST_INSTANCE_SECTION command is set to SELECTED (the default), the instance section contains instances connected to extracted nets.
- If the command is set to YES, the instance section contains all instances.
- If the command is set to NO, the instance section is not generated.

Examples

The following commands extract nets that begin with A or B:

```
NETS: A*
NETS: B*
```

The following commands extract nets that begin with A, but no nets beginning with AD unless they begin with ADQ:

```
NETS: A*
NETS: !AD*
NETS: ADQ*
```

Verilog Example

The following example extracts names from hierarchical Verilog and assumes that the place and route tool specified to handle special characters in the Verilog identifiers.

```
module low ( A , Y ) ;
   input A ;
   output Y ;
   wire n1 ;
   INVX1  X0  (.A(A ), .Y(n1 )) ;
   INVX1  X1  (.A(n1 ), .Y(Y ));
endmodule

module mid ( IN , OUT ) ;
   input IN ;
   output OUT ;
   wire \instA/din[1] , net2 ;
   low  U0/instA  (.A(IN ), .Y(\instA/din[1] )) ;
   INVX1  U1  (.A(\instA/din[1] ), .Y(net2 )) ;
   INVX1  U2  (.A(net2 ), .Y(OUT )) ;
endmodule

module top ( SIG, SAME ) ;
   input SIG ;
   output SAME ;
   wire route1 ;
   mid  U10  (.IN(SIG ), .OUT(route1 ));
   mid  U11  (.IN(route1 ), .OUT(SAME ));
endmodule
```

Assume that the `HIERARCHICAL_SEPARATOR:/` and `BUS_BIT: []` commands are used. To extract `instA/din[1]` from instance `U10` in block `top`, specify the following:

```
NETS:  U10/instA\/din\[1\]
```

To extract `n1` from `U0/instA` of `U10` in block `top`, specify the following:

```
NETS:  U10/U0\/instA/n1
```

See Also

- BUS_BIT
- CASE_SENSITIVE
- HIERARCHICAL_SEPARATOR

- NET_TYPE
- NETS_FILE

Feedback

NETS_FILE

Specifies one or more files containing a series of NETS commands.

Syntax

```
NETS_FILE: file1 [file2… fileN]
```

Arguments

Argument	Description
file1 [*file2… fileN*]	Files containing the NETS command lines

Description

This command specifies one or more files containing a series of NETS commands.

Examples

In this example, the myNets file contains the following lines:

```
NETS: neta1 neta2
NETS: netb1 netb2 netb3
NETS: clock1
```

The following command specifies that the myNets file contains the series of NETS commands:

```
NETS_FILE: myNets
```

See Also

- BUS_BIT
- CASE_SENSITIVE
- HIERARCHICAL_SEPARATOR
- NETS

Feedback

NODENAME_NETNAME_ON_DANGLING_PORTS

Retains net names for nets with dangling ports. Valid only for transistor-level GPD flows.

Syntax

```
NODENAME_NETNAME_ON_DANGLING_PORTS: YES | NO
```

Arguments

Argument	Description
YES (default)	For dangling ports, creates additional nodes and retains net names for those nodes
NO	Does not retain subnode names for dangling ports

Description

The NODENAME_NETNAME_ON_DANGLING_PORTS command has an effect only if you set the NETLIST_NODENAME_NETNAME command to YES. For more information about the NETLIST_NODENAME_NETNAME command behavior, see the reference page.

For a net with a dangling port (a net with only one *|I node), the StarRC tool adds a new node shorted to the *|I node and renames the newly created node to the net name.

This command is valid only for transistor-level GPD flows and is ignored in other flows.

See Also

- NETLIST_FORMAT
- NETLIST_NODENAME_NETNAME

Feedback

NON_COLOR_POLYGON_HANDLING

Specifies how to treat polygons that do not have color assignments in a multiple patterning layer.

Syntax

```
NON_COLOR_POLYGON_HANDLING: NONE | DROP | WARN | DROP_AND_WARN | ERROR |
 ERROR_EXCEPT_FULL_OVERLAP
```

Arguments

Argument	Description
NONE	Translates noncolored polygons without issuing a warning message
DROP	Drops (does not translate) all noncolored polygons
WARN	Translates noncolored polygons, issues a warning message, and continues the run
DROP_AND_WARN	Drops (does not translate) all noncolored polygons, issues a warning message, and continues the run
ERROR	Stops the extraction and issues an error message
ERROR_EXCEPT_FULL_OVERLAP	Drops (does not translate) each noncolored polygon that is fully covered with colored polygons on the same layer without issuing a warning message Stops extraction and issues an error message if any noncolored polygon that is not fully covered by colored polygons exists on the same layer

Description

In a multiple patterning layer, most polygons have a color assignment (mask assignment) to specify which mask should contain that polygon. The NON_COLOR_POLYGON_HANDLING command specifies the action that the StarRC tool should take if it encounters a polygon without a color assignment on a multiple patterning layer. The tool issues the following error message when there are polygons without color assignment in the design:

```
ERROR: Found non-color polygon on multi-patterning Layer
'%s'. BBox=(%g,%g),(%g,%g) %s
```

The NON_COLOR_POLYGON_HANDLING NONE or NON_COLOR_POLYGON_HANDLING DROP command issues the following message when a polygon without color assignment is detected in a layer assigned with the NUMBER_OF_MASKS and MASKS keywords.

```
WARNING: Polygon without MASK info detected on a layer with MASK info in
   ITF; Please check StarRC User Guide and run with
```

```
NON_COLOR_POLYGON_HANDLING: DROP_AND_WARN to get list of polygons
without MASK information.(SX-3931)
```

The default for the NON_COLOR_POLYGON_HANDLING command changes based on process nodes and their versions. For information about the default settings of the command, contact your Synopsys support representative.

For all Self-Aligned Double Patterning (SADP) process, use the NON_COLOR_POLYGON_HANDLING: WARN | ERROR command

For signoff extraction, use the NON_COLOR_POLYGON_HANDLING: ERROR command.

Feedback

NONCRITICAL_COUPLING_REPORT_FILE

Specifies the file name for the noncritical coupling report.

Syntax

```
NONCRITICAL_COUPLING_REPORT_FILE: output_file
```

Arguments

Argument	Description
output_file	File name for the noncritical coupling report file Default: none

Description

Report file generation is supported in the Milkyway flow. The format of the file includes the following:

- A comment at the top of the file refers to the corresponding SPEF file name, *prefix* command, and *suffix* command.
- The report lists all coupling capacitances from noncritical nets to critical nodes, in reverse order from the .spef output. For example, if the SPEF lists the first line shown in the following, the report output lists what is shown on the second line.

```
SPEF:
14 A B/SYNOPSYS_INCONTEXT_b 1.0e-15

Report file:
14 B/SYNOPSYS_INCONTEXT_b A 1.0e-15
```

- The command works with `NETLIST_NAME_MAP:YES | NO` for net name mapping of noncritical net names.

Do not specify the same name for the report file name with either the `NETLIST_FILE` or `COUPLING_REPORT_FILE` commands. Otherwise, the StarRC tool generates an error message and stops.

Retaining coupling capacitances between the top-level parent routing and the skip cell child net routing exists for the Milkyway flow using the SPEF netlist format.

Only SPEF format is supported. If the user-specified netlist is not SPEF, the tool issues a warning message.

Feedback

See Also

- NETLIST_FORMAT : SPEF
- COUPLE_NONCRITICAL_NETS
- COUPLE_NONCRITICAL_NETS_PREFIX
- RING_AROUND_THE_BLOCK
- SKIP_CELLS_COUPLE_TO_NET
- ZONE_COUPLE_TO_NET

Feedback

NUM_CORES

Specifies the number of cores used for distributed processing.

Syntax

```
NUM_CORES: number_of_cores
```

Arguments

Argument	Description
number_of_cores	The number of cores Default: 1

Description

The NUM_CORES command enables distributed processing for extraction and specifies the number of cores to use.

If you specify a value greater than the number of cores available, the StarRC and grdgenxo tools issue a warning and use as many cores as available.

For more information, see Distributed Processing and Using Distributed Processing With the grdgenxo Tool.

Note:

The number of worker processes launched by the StarRC and grdgenxo tools is equal to the setting of the NUM_CORES command. If your submission command specifies a smaller number of cores, some cores are reserved but not used. For best results, the NUM_CORES command and the submission command should specify the same number of cores.

See Also

- STARRC_DP_STRING
- GRD_DP_STRING

Feedback

NUM_CORES_FOR_HIGH_MEM_TASKS

Specifies number of cores to handle high memory jobs.

Syntax

```
NUM_CORES_FOR_HIGH_MEM_TASKS: number_of_cores
```

Arguments

Argument	Description
number_of_cores	Number of cores to be used for high memory jobs Default: 1 Maximum: 5

Description

The NUM_CORES_FOR_HIGH_MEM_TASKS command enables distributed processing specifically for tasks or jobs that consume high memory. The maximum number of cores that you can specify is 5.

If you specify a value greater than the available number of cores, then extraction begins with the available number of cores only. If both NUM_CORES and NUM_CORES_FOR_HIGH_MEM_TASKS commands are specified in the command file, the total requested cores for the extraction is the sum of the cores specified with the NUM_CORES and NUM_CORES_FOR_HIGH_MEM_TASKS commands.

Cores meant for high memory tasks are also used for regular tasks if high memory task is not available for processing. The NUM_CORES_FOR_HIGH_MEM_TASKS command must be used along with the STARRC_DP_STRING_FOR_HIGH_MEM_TASKS command.

See Also

- STARRC_DP_STRING
- STARRC_DP_STRING_FOR_HIGH_MEM_TASKS

Feedback

OA_BUS_BIT

Specifies the delimiters used during OpenAccess view creation for bus bits in transistor-level flows.

Syntax

```
OA_BUS_BIT: {} | [] | () | <>
```

Arguments

Argument	Description
{}[]()<>	Characters used as the bus bit or iterated instance delimiters; do not insert spaces between the characters in the string Default: the delimiters specified by the BUS_BIT command

Description

The OA_BUS_BIT command specifies the delimiters used during OA view creation for bus bits. Use this command if you need different bus bit delimiters than those specified by the BUS_BIT command for the original database and ASCII flow. The OA_BUS_BIT command applies to view creation, port annotation, and schematic view annotation.

You must use the BUS_BIT command to specify the bus bit delimiters for the original database before using the OA_BUS_BIT command. If you use the OA_BUS_BIT command alone, the StarRC tool issues a warning message and does not change the bus bit delimiters in the OpenAccess view.

Examples

In the following example, bus name a(2) becomes a<2> in the OA view:

```
BUS_BIT: ()
OA_BUS_BIT: <>
```

See Also

- BUS_BIT
- OA_INSTANCE_BIT

Feedback

OA_CARRY_SCH_MODEL_NAME

Specifies whether to use schematic model names in the OA view if the schematic model names are different from the LVS names.

Syntax

```
OA_CARRY_SCH_MODEL_NAME: YES | NO
```

Arguments

Argument	Description
YES	Use schematic model names
NO (default)	Use default model names

Description

The OA_CARRY_SCH_MODEL_NAME command specifies whether the StarRC tool should use model names from the schematic in the OA view if the schematic model names are different from the names in the layout versus schematic output.

In some designs, a schematic might have different device model names even when those devices have the same model name in the LVS runset. For accurate simulation in downstream tools, set the OA_CARRY_SCH_MODEL_NAME command to YES to provide the correct model name in the output netlist.

See Also

- RCGenParaViewBatch

Feedback

OA_CDLOUT_RUNDIR

Specifies the path to the CNL or CDLout directory created by a custom design tool.

Syntax

```
OA_CDLOUT_RUNDIR: cdl_dir
```

Arguments

Argument	Description
cdl_dir	Path to the directory Default: none

Description

This command specifies the path to a directory created by a custom design tool that contains mapping files necessary for correct cross-referencing between SPICE and OpenAccess netlists. The goal is to obtain the original schematic names of nets in the OpenAccess view, because some names might have changed during OpenAccess netlist creation.

The argument can be either an absolute path or relative path (relative to the working directory). The path should not include the cnl or ihnl directory name, but instead should end with the parent directory of the cnl or ihnl directory. Ending the path with a slash character is acceptable but not required.

The StarRC tool can read two types of directories:

- CNL directory created by the Custom Compiler tool

  The StarRC tool reads the XML format name mapping files.

- CDL out directory created by the Virtuoso tool

  The StarRC tool reads the ASCII format files found in the ihnl subdirectory.

The StarRC tool reads information in the files related to schematic names and the design hierarchy; all other information is ignored.

Examples

```
OA_CDLOUT_RUNDIR: /eng_svr10/designs/cpu/lvs/
```

See Also

- NETLIST_FORMAT : OA

Feedback

OA_CELL_NAME

Specifies the name of the OpenAccess (OA) cell for which the parasitic view is written.

Syntax

```
OA_CELL_NAME: cell_name
```

Arguments

Argument	Description
cell_name	Cell name Default: the name specified in the BLOCK command

Description

The OA_CELL_NAME command specifies the name of the OpenAccess (OA) cell for which the parasitic view is written.

Examples

```
OA_CELL_NAME: vco
```

See Also

- OA_LIB_NAME

Feedback

OA_DEVICE_MAPPING_FILE

Specifies the mapping file that links database and OpenAccess device names.

Syntax

```
OA_DEVICE_MAPPING_FILE: map_file
```

Arguments

Argument	Description
map_file	OA device mapping file location Default: none

Description

The `OA_DEVICE_MAPPING_FILE` command specifies a device mapping file that maps ideal and parasitic devices in the StarRC parasitic output to the corresponding device symbols in the OpenAccess symbol libraries. This file contains an entry for every ideal and parasitic device model that exists in the parasitic output. It also provides the ability to remap standard StarRC DSPF device property names to user-specified property names.

This command accepts files in the yaml format in addition to the formats described in Chapter 7, Using StarRC With the Custom Compiler Tool and Chapter 8, Using StarRC With the Virtuoso Tool.

Examples

```
OA_DEVICE_MAPPING_FILE: /path/oa_device_map
```

See Also

- NETLIST_FORMAT : OA
- OA_INSTANCE_PIN_NAME
- OA_LAYER_MAPPING_FILE

Feedback

OA_INSTANCE_BIT

Specifies the delimiters used during OpenAccess view creation for iterated instances in transistor-level flows.

Syntax

```
OA_INSTANCE_BIT: {} | [] | () | <>
```

Arguments

Argument	Description
{}[]()<>	Characters used as the iterated instance delimiters; do not insert spaces between the characters in the string Default: the delimiters specified by the OA_BUS_BIT command

Description

The OA_INSTANCE_BIT command specifies the delimiters used during OpenAccess (OA) view creation for iterated instances, which are also known as arrayed instances or instance vectors. Use this command if you need different delimiters than those specified by the BUS_BIT or OA_BUS_BIT commands for the original database and ASCII flow. The OA_INSTANCE_BIT command applies to view creation, port annotation, and schematic view annotation.

You must use the BUS_BIT command to specify the bus bit delimiters for the original database before using the OA_BUS_BIT and OA_INSTANCE_BIT commands. If you use the OA_BUS_BIT or OA_INSTANCE_BIT command alone, the StarRC tool issues a warning message and does not change the delimiters in the OpenAccess view.

Examples

In the following example, bus name a[2] becomes a<2> in the OA view, but instance name IO[3] remains as IO[3]:

```
BUS_BIT: []
OA_BUS_BIT: <>
OA_INSTANCE_BIT: []
```

See Also

- BUS_BIT
- OA_BUS_BIT

Feedback

OA_INSTANCE_PIN_NAME

Specifies whether the StarRC parasitic view obtains pin names from the OpenAccess device mapping file.

Syntax

```
OA_INSTANCE_PIN_NAME: SYMBOL | SUBCKT
```

Arguments

Argument	Description
SYMBOL	Use pin names defined in the OpenAccess device mapping file
SUBCKT (default)	Use pin names from the layout versus schematic tool

Description

The OA_INSTANCE_PIN_NAME command specifies which pin names to use in the StarRC view. By default, pin names come from the layout versus schematic (LVS) tool. If you set the OA_INSTANCE_PIN_NAME command to SYMBOL, pin names come from the device mapping file specified by the OA_DEVICE_MAPPING_FILE command.

See Also

- OA_DEVICE_MAPPING_FILE

Feedback

OA_LAYER_MAPPING_FILE

Specifies the mapping file to link database and OA layer names.

Syntax

```
OA_LAYER_MAPPING_FILE: file_path
```

Arguments

Argument	Description
file_path	OA layer mapping file location Default: none

Description

The `OA_LAYER_MAPPING_FILE` command specifies a layer mapping file that maps StarRC runset layers from the StarRC mapping file to the corresponding OpenAccess technology file layers. This allows polygons, ports, and subnodes from the parasitic extraction to be stored within the StarRC generated OA parasitic view.

Examples

```
OA_LAYER_MAPPING_FILE: /path/oa_layer_map
```

See Also

- NETLIST_FORMAT : OA
- OA_DEVICE_MAPPING_FILE
- OpenAccess Mapping Files

Feedback

OA_LIB_DEF

Specifies the path to a file that defines libraries referenced in OA_DEVICE_MAPPING_FILE and OA_LAYER_MAPPING_FILE.

Syntax

```
OA_LIB_DEF: file_path
```

Arguments

Argument	Description
file_path	Path to the library definition file Default: lib.defs

Description

The OA_LIB_DEF command specifies the file name, including the full path, that defines libraries referenced by any of the following commands:

- OA_DEVICE_MAPPING_FILE
- OA_LAYER_MAPPING_FILE
- OA_LIB_NAME
- OA_SKIPCELL_MAPPING_FILE
- OA_PORT_ANNOTATION_VIEW
- OA_PROPERTY_ANNOTATION_VIEW

Examples

```
OA_LIB_DEF: /path/lib.def
```

See Also

- OA_DEVICE_MAPPING_FILE
- OA_LAYER_MAPPING_FILE
- OA_LIB_NAME
- OA_SKIPCELL_MAPPING_FILE
- OA_PORT_ANNOTATION_VIEW
- OA_PROPERTY_ANNOTATION_VIEW

Feedback

OA_LIB_NAME

Specifies the name of the OpenAccess (OA) library.

Syntax

```
OA_LIB_NAME: library_name
```

Arguments

Argument	Description
library_name	Library name Default: none

Description

The OA_LIB_NAME command specifies the name of the OpenAccess (OA) library. The path to the library can be specified in the OA_LIB_DEF file.

Examples

```
OA_LIB_NAME: PLL
```

See Also

- NETLIST_FORMAT : OA
- OA_DEVICE_MAPPING_FILE
- OA_LAYER_MAPPING_FILE

Feedback

OA_MARKER_SIZE

Specifies the port or subnode marker size.

Syntax

```
OA_MARKER_SIZE: value
```

Arguments

Argument	Description
value	The port or subnode marker size Units: microns Default: 0.1

Description

This command is optional.

Examples

```
OA_MARKER_SIZE: 0.4
```

See Also

- NETLIST_FORMAT : OA

Feedback

OA_MULTI_OUTPUT

Creates one additional parasitic netlist when OpenAccess output is generated. Valid only for transistor-level IC Validator and Calibre Connectivity Interface flows.

Syntax

```
OA_MULTI_OUTPUT: SPF | SPEF | STAR
```

Arguments

Argument	Description
SPF (default)	Creates an SPF netlist in addition to the OA netlist
SPEF	Creates an SPEF netlist in addition to the OA netlist
STAR	Creates a STAR format netlist in addition to the OA netlist

Description

The OA_MULTI_OUTPUT command allows you to specify the format of a parasitic netlist to be created at the same time as an OpenAccess parasitic view netlist.

See Also

- OA_DEVICE_MAPPING_FILE
- NETLIST_FORMAT

Feedback

OA_NOT_GLOBAL_NETS

Specifies whether the ground node is set to global for OpenAccess views.

Syntax

```
OA_NOT_GLOBAL_NETS: ground_node_name
```

Arguments

Argument	Description
ground_node_name	Ground node Default: !*

Description

By default, the ground node identified by the NETLIST_GROUND_NODE_NAME command is set as a global node in the OpenAccess (OA) parasitic view. The default ground node name is 0.

To prevent setting the ground node as a global node, specify the ground node name with the OA_NOT_GLOBAL_NETS command.

Examples

```
OA_NOT_GLOBAL_NETS: 0
```

See Also

- NETLIST_GROUND_NODE_NAME

Feedback

OA_OVERWRITE_LOCKED_VIEW

Specifies whether the StarRC tool can overwrite a locked parasitic view in the Virtuoso Integration flow.

Syntax

```
OA_OVERWRITE_LOCKED_VIEW: YES | NO
```

Arguments

Argument	Description
YES	Overwriting is allowed
NO (default)	Overwriting is not allowed

Description

The OA_OVERWRITE_LOCKED_VIEW command specifies whether the StarRC tool can overwrite a locked OpenAccess parasitic view in the Virtuoso Integration interface. A parasitic view is locked if it is opened for editing while StarRC is running.

If the command is set to NO (the default), the tool issues an error message if you try to save a parasitic view that is already locked.

You can also enable this feature by using the overwrite_locked_view key in the RCGenParaViewBatch procedure.

See Also

- RCGenParaViewBatch

Feedback

OA_PORT_ANNOTATION_VIEW

Enables the simulation of a parasitic view generated by the OpenAccess writer.

Syntax

```
OA_PORT_ANNOTATION_VIEW: lib_name cell_name view_name
```

Arguments

Argument	Description
lib_name	Library name containing the top-level port information
cell_name	Cell name containing the top-level port information
view_name	View name containing the top-level port information

Description

Use this command to specify a library name, a cell name, and a view name to define the pins or ports to be included in the output parasitic OA view. The specified library, cell, or view name must correspond to the top-level block. If the OpenAccess writer cannot find the named object, a warning is issued and the annotation view is not generated.

The command functions as follows:

- For each port in the `OA_PORT_ANNOTATION_VIEW` argument list, there must be a port on the net in the parasitic view with the same name. If the net does not have that same port, a port is created on that net.
- If the port has been created after extraction, there is no annotation for that port.

Examples

The following command specifies that the schematic view from library alib and cell acell should be used to determine the ports to be included in the output parasitic OA view:

```
OA_PORT_ANNOTATION_VIEW: alib acell symbol
```

See Also

- OA_PROPERTY_ANNOTATION_VIEW

Feedback

OA_PROPERTY_ANNOTATION_VIEW

Specifies a schematic library, cell, or view to check against ideal devices for schematic-only properties and to attach them to the OpenAccess parasitic view.

Syntax

```
OA_PROPERTY_ANNOTATION_VIEW: [lib] [cell] view_name
```

Arguments

Argument	Description
`lib`	Checks ideal devices for schematic-only properties in this library
`cell`	Checks ideal devices for schematic-only properties in this cell
`view_name`	A view name in the current extraction library. It can be a library name, cell name, or a view name. A warning is issued - If the specified library, cell, or view cannot be found (cannot be opened). - If you specify more than the allowed names (an invalid value).

Description

Specifies a schematic library, cell, or view to check against ideal devices for schematic-only properties and to attach them to the OpenAccess parasitic view.

If an ideal instance cannot be correctly cross-referenced, the OpenAccess writer does not execute a schematic annotation and issues a warning message similar to the following:

```
Warning: Instance I0|I1|ld_M21 can't get property from schematic view as
 not-XREF-ed
```

Only `XREF:YES` and `XREF:COMPLETE` are supported in the schematic view annotation.

Examples

The following command specifies that alib/acell/schematic (library/cell/view) is checked for schematic-only properties of ideal instances. The properties are then copied to the corresponding ideal instances in the OpenAccess parasitic view created by the StarRC tool.

```
OA_PROPERTY_ANNOTATION_VIEW: alib acell schematic
```

See Also

- XREF
- OA_PORT_ANNOTATION_VIEW

Feedback

OA_PROPMAP_CASE_SENSITIVE

Specifies whether the schematic view property annotation is case-sensitive.

Syntax

```
OA_PROPMAP_CASE_SENSITIVE : YES | NO | MIXED
```

Arguments

Argument	Description
YES	Property names are kept unchanged
NO (default)	Property names are converted to lowercase
MIXED	Property names are kept unchanged except for specific exceptions, which are converted to lowercase

Description

The OA_PROPMAP_CASE_SENSITIVE command affects schematic property annotation with the StarRC OpenAccess file writer, as follows:

- A setting of YES or t means that all property names are kept unchanged.
- A setting of NO or nil means that all property names are converted to lowercase.
- A setting of MIXED means that property names are kept unchanged except for the following property names, which are converted to lowercase: l, w, as, ad, ps, pd, nrd, nrs, m, area, and pj. This is the recommended setting for mixed-case property names.

The OA_PROPMAP_CASE_SENSITIVE command is similar to the PROPMAP_CASE_SENSITIVE option in the .snps_settings file used for the Virtuoso Integration flow, but the OA_PROPMAP_CASE_SENSITIVE command is used for running the StarRC tool as a standalone tool.

See Also

- PROPMAP Case Sensitivity

Feedback

OA_REMOVE_DUPLICATE_PORTS

Prevents duplication when annotating both the vector-form port names and the expanded port names.

Syntax

```
OA_REMOVE_DUPLICATE_PORTS: YES | NO
```

Arguments

Argument	Description
YES	Avoids duplicate ports in the OA view
NO (default)	Annotates all the ports in the OA view

Description

When your design has nonexpanded port names in the schematic view and expanded BUS port names in the layout, the StarRC tool annotates both port names by default. If you do not want this duplication, set the OA_REMOVE_DUPLICATE_PORTS command to YES.

Examples

If you have OTP<0:1> for the nonexpanded port names in the schematic view and then have OTP<0> and OTP<1> in the layout view, by default, the tool outputs all three ports in the final OA parasitic view:

OTP<0:1> , OTP<0> , OTP<1>

When you set the OA_REMOVE_DUPLICATE_PORTS command to YES, the tool only writes the OTP<0> and OTP<1> ports in the netlist.

See Also

- OA_PORT_ANNOTATION_VIEW

Feedback

OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX

Specifies whether SPICE card prefix characters appearing before the primitives of instance names should be removed.

Syntax

```
OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX: YES | NO | cdl_file
```

Arguments

Argument	Description
`YES` (default)	Removes the SPICE card prefix characters from the primitives of instance names
`NO`	Preserves the SPICE card prefix characters in the primitives of instance names
`cdl_file`	The file name of a cdlinclude file that contains the SUBCKT definitions for which the primitive X card should be stripped

Description

The `OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX` command specifies whether to remove SPICE card prefix characters before the primitives in ideal instance names. Prefix characters are C, D, M, R, Q, or X.

You can optionally provide the name of a cdlinclude file that contains SUBCKT definitions.

A ?spiceCardStripPrimitiveCallBack key in a SKILL procedure takes precedence over a `OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX` command in a StarRC command file.

Examples

If the layout-versus-schematic tool provides instance names as follows:

```
XXI18/XI0/M1
XXI18/XI1/R1
```

After SPICE card prefix removal, the instance names become:

```
XXI18|XI0|1
XXI18|XI1|1
```

See Also

- OA_REMOVE_SPICECARD_PREFIX

Feedback

OA_REMOVE_SPICECARD_PREFIX

For Virtuoso Integration flows, specifies whether the SPICE card prefix in the paths of ideal instance names and net names should be removed.

Syntax

```
OA_REMOVE_SPICECARD_PREFIX: YES | NO
```

Arguments

Argument	Description
`YES` (default)	Removes the prefix characters from the paths of instance names and net names
`NO`	Preserves the prefix characters in the paths of instance names and net names

Description

The `OA_REMOVE_SPICECARD_PREFIX` command specifies whether to remove SPICE card prefix characters from the paths of ideal instance names. Prefix characters are C, D, M, R, Q, or X.

A ?spiceCardStripinstpathCallBack key in a SKILL procedure takes precedence over the `OA_REMOVE_SPICECARD_PREFIX` command.

Examples

If the layout-versus-schematic tool provides instance names as follows:

```
XXI18/XI0/M1
XXI18/XI1/R1
```

After SPICE card prefix removal, the instance names become:

```
I18|I0|M1
I18|I1|R1
```

See Also

- OA_REMOVE_PRIMITIVE_SPICECARD_PREFIX

Feedback

OA_SCHEMATIC_PCELL_EVAL_LIBRARY

Defines an OpenAccess (OA) library path for evaluation of properties from schematic PCells.

Syntax

```
OA_SCHEMATIC_PCELL_EVAL_LIBRARY: library_path
```

Arguments

Argument	Description
library_path	Specified library path Default: none

Description

When the `OA_PROPERTY_ANNOTATION_VIEW` command is set to `YES` or when the `? carry_sch_property t` command is specified in the RCGenParaViewBatch procedure, the StarRC tool checks ideal devices for schematic-only properties and attaches them to the OpenAccess parasitic view. If the design contains schematic PCells, the PCells must be evaluated to determine their properties. The `OA_SCHEMATIC_PCELL_EVAL_LIBRARY` command specifies a library to use for PCell property evaluation.

By default, if this command is not used and the underlying SKILL code is available, the StarRC tool creates a library named StarRCPCellLib in the current working directory.

See Also

- OA_PROPERTY_ANNOTATION_VIEW

Feedback

OA_SEPARATE_PARASITICS

Creates compact size of the OpenAccess parasitic view in a GPD flow.

Syntax

```
OA_SEPARATE_PARASITICS: YES | No
```

Arguments

Argument	Description
YES	Creates compact size of the OA parasitic view
NO	Does not create the compact size of the OA parasitic view

Description

The `OA_SEPARATE_PARASITICS: YES` command creates compact size of the OA parasitics view for parasitics resistors and capacitors only and provides the following information

- Displays canonical devices from the LVS tool only in the resulting OA view.
- Writes parasitics devices to a separate SPF file.

Both canonical and parasitic devices are put together in an output netlist during simulation.

Note:

This command is not effective when used on its own, so you must use the command along with other `OA_*` commands.

Feedback

OA_SKIPCELL_MAPPING_FILE

Specifies a file to define skip cell extraction in an OpenAccess flow.

Syntax

```
OA_SKIPCELL_MAPPING_FILE: oa_skip_file
```

Arguments

Argument	Description
oa_skip_file	Name of the cell to skip at a particular hierarchical level Default: none

Description

For hierarchical designs, you might only want to extract the top-level design for a parasitic view without extracting the lower-level block to reduce the view generation time. In the ASCII DSPF flow, the SKIP_CELLS command is typically used for pre-extracted blocks. In a DSPF flow, those skip cells specified in a SKIP_CELLS command are listed in the *Instance" section for simulation purposes.

To enable the skipping operation in an OpenAccess parasitic view, you must specify which cell master to use for the skip cell instantiation in the parasitic view. Each specified skip cell has corresponding mapping information for cell instantiation in the parasitic view.

Examples

```
SKIP_CELLS:  INV1  INV2
OA_SKIPCELL_MAPPING_FILE: skip_file
```

File skip_file contains the following:

```
INV1  analogLib INV symbol
INV2  analogLib INV symbol
INV3  analogLib INV symbol
```

Even though there might be an *INV3* in the top block, it is not treated as a skip cell because it is not listed in the SKIP_CELLS command.

See Also

- NETLIST_FORMAT : OA
- SKIP_CELLS

Feedback

OA_VIEW_NAME

Specifies the name of the OpenAccess parasitic view.

Syntax

```
OA_VIEW_NAME: view_name
```

Arguments

Argument	Description
view_name	Name of OA parasitic view Default: starrc

Description

You can specify the name of the OA parasitic view. By default, the OA parasitic view name is starrc. This command is optional.

Examples

```
OA_VIEW_NAME: extract_view
```

See Also

- NETLIST_FORMAT : OA

Feedback

OASIS_FILE

Specifies OASIS® format files to represent part of the physical layout.

Syntax

```
OASIS_FILE: file1 [file2] …
```

Arguments

Argument	Description
file1 [file2] …	OASIS file names Default: none

Description

The OASIS_FILE command specifies OASIS format files to represent part of the physical layout.

In the Fusion Compiler or IC Compiler II flow (NDM format designs) and IC Compiler flow (Milkyway designs), this command merges OASIS data into FRAM or CEL views for skip cells to provide a full physical layout representation. For the cells that are defined by both the OASIS file and the FRAM or CEL views, the StarRC tool uses only the pin shapes from the FRAM or CEL views. StarRC replaces any cells in the obstruction section of the FRAM or CEL view with cells of the same name in the OASIS file. If the tool does not find a matching cell name in the OASIS file, then the FRAM or CEL view continues to be used for that cell.

For LEF/DEF designs, this command merges OASIS data into LEF MACRO definitions for skip cells to provide a full physical layout representation. For the cells that are defined by both the OASIS file and the LEF macro, the StarRC tool uses only the pin shapes from the LEF macro. StarRC replaces any cells in the OBS section of the LEF MACRO with cells of the same name in the OASIS file. If the tool does not find a matching cell name in the OASIS file for a particular DEF cell placement, then the OBS section of the LEF MACRO continues to be used for that cell.

The OASIS_FILE command can be specified multiple times. It must be used with the OASIS_LAYER_MAP_FILE command and cannot be used with the GDS_FILE command.

See Also

- OASIS_LAYER_MAP_FILE
- SKIP_CELLS

Feedback

OASIS_LAYER_MAP_FILE

Specifies the mapping between the OASIS® layer number and layer name in the design database.

Syntax

```
OASIS_LAYER_MAP_FILE: file_name
```

Arguments

Argument	Description
file_name	OASIS layer mapping file name Default: none

Description

The OASIS_LAYER_MAP_FILE command specifies the mapping between the OASIS layer number and layer name in the design database whenever OASIS_FILE or METAL_FILL_OASIS_FILE is used to import OASIS data into the design database.

Note:

You cannot use the OASIS_LAYER_MAP_FILE command together with the GDS_LAYER_MAP_FILE command.

The OASIS layer map file uses the following syntax:

```
database_layer oasis_layer_number oasis_datatype [MASK=mask_no]
           {FLOATING | GROUNDED | IGNORE} [IP_FILL]
```

Argument	Description
database_layer	The database layer name.
oasis_layer_number	The OASIS layer number.
oasis_datatype	The OASIS data type. If a data type is not specified, all data types on a given layer are read.
mask_no	The mask ID number for multimask patterning (an integer).
FLOATING	Specifies that the corresponding fill layer is to be treated as floating. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.

Feedback

Argument	Description
GROUNDED	Specifies that the corresponding fill layer is to be treated as grounded. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.
IGNORE	Specifies that the corresponding fill layer is to be ignored. Valid if the METAL_FILL_POLYGON_HANDLING: AUTOMATIC command is used. Otherwise, this setting is ignored.
IP_FILL	Differentiates metal fills from other design information in the OASIS layer map.

All OASIS layers for translation must have an entry in this file and must have a definition in the layout database.

Use the OASIS_LAYER_MAP_FILE command with the following commands:

- OASIS_FILE, when importing OASIS cells into a Milkyway or LEF/DEF database
- METAL_FILL_OASIS_FILE, when importing metal fill polygons into a Milkyway, LEF/DEF, or Calibre Connectivity Interface database

If you use both the METAL_FILL_OASIS_FILE and OASIS_FILE commands in a single Milkyway or LEF/DEF run, you should use a single unified layer mapping file for both the OASIS files.

If a layer in the mapping file does not have a corresponding layer in the layout database, the StarRC tool issues an error message.

The additional specification of a layer-specific fill-handling keyword is available to allow users to selectively decide how individual metal fill layers are handled during parasitic extraction. This handling is considered only when METAL_FILL_POLYGON_HANDLING: AUTOMATIC is set in the StarRC command file. If METAL_FILL_POLYGON_HANDLING: AUTOMATIC is set in the command file, but a fill-handling mode is not specified for a particular layer inside OASIS_LAYER_MAP_FILE, the setting defaults to FLOATING for that layer. If another setting of METAL_FILL_POLYGON_HANDLING (other than AUTOMATIC) is specified in the command file, that setting governs the handling of all layers, and any layer-specific mode specifications inside OASIS_LAYER_MAP_FILE are disregarded.

Note that in the given example, handling mode GROUNDED is specified for layer DIFF. If METAL_FILL_POLYGON_HANDLING: AUTOMATIC is also specified in the command file, DIFF is treated as GROUNDED while all other layers are treated as FLOATING. If METAL_FILL_POLYGON_HANDLING:AUTOMATIC is not specified, the layer-specific mode specification for DIFF is disregarded, and the global mode set by METAL_FILL_POLYGON_HANDLING takes precedence.

Feedback

Examples

The following example shows how the DIFF layer is assigned to OASIS layer 2 and OASIS datatype 0. Note that this example maps layers from a `METAL_FILL_OASIS_FILE` and specifies layer-specific fill handling for the DIFF layer.

Example 32 Layer-Specific Fill Handling

```
DIFF 2 0      GROUNDED
POLY 7 0
CONT 4 0
METAL1 10 0
METAL1 10 1
METAL1 76 0
VIA1 11 0
METAL2 12 0
```

In the following example, the `METAL_FILL_POLYGON_HANDLING: AUTOMATIC` command is set in the command file.

Example 33 Automatic Fill Handling

```
*layer treated as grounded
DIFF    2 0 GROUNDED
*layer treated as floating
POLY    7 0 FLOATING
*layer governed by default floating mode since mode is unspecified.
METAL1    4 0
```

See Also

- OASIS_FILE
- LEF_FILE
- METAL_FILL_OASIS_FILE

Feedback

OBSERVATION_POINTS

Specifies cells that are not skipped for reporting observation nodes in the output netlist.

Syntax

```
OBSERVATION_POINTS: cell_list
```

Arguments

Argument	Description
cell-list	List of nonskipped cell names; wildcards allowed Default: none

Description

This command generates nodes at nonskipped hierarchical interactions. Observation nodes are reported in the parasitics section of the netlist, allowing these locations to be used during simulation. This command is not supported for LEF/DEF input.

Observation nodes are written as **O statements in SPEF netlists and as **|O statements in SPF, NETNAME, and STAR format netlists. The format of observation point statements is identical to that of *I and *|I statements.

The StarRC tool treats observation points consistently between gate-level and transistor-level extraction, as follows:

- The **|O location in the netlist for a PCell is always reported at the top-level interconnect layer.
- If the PCell connects to the top-level interconnect layer in multiple locations, the netlist contains multiple **|O lines.
- If a PCell does not connect to the top-level interconnect layer, the reported PCell observation point is the lowest-left coordinate on the highest layer present in the PCell.

The OBSERVATION_POINTS command works with the XREF:NO, YES, and COMPLETE commands. Only EQUIV points can be selected cells with the OBSERVATION_POINTS command. Schematic names are used with observation nodes in the netlist when the XREF command is used. The CELL_TYPE command determines which cell names are applied for selection.

Note:

If the XREF command is set to YES, the netlist contains **|O lines only when PCells in the layout have an equivalent hierarchy in the schematic.

Feedback

Observation nodes are formed from marker layers. If there are no marker shapes in a selected level of hierarchy, there are no **O statements. With the default automatic marker generation, there must be a hierarchical interaction of the routing layers to get a marker shape for **O at that level of the hierarchy. You can also control observation points by generating marker shapes with the `MARKER_GENERATION: USER` command.

The marker shape is not related to `TEXT_POLYGON` commands in the Hercules runset or marker layers in the mapping file when `MARKER_GENERATION` is set to `AUTOMATIC`.

See Also

- CELL_TYPE
- MARKER_GENERATION
- XREF
- Parameterized Cells

Feedback

OPERATING_FREQUENCY

Specifies the frequency used when including substrate effects during high frequency extractions.

Syntax

```
OPERATING_FREQUENCY: freq
```

Arguments

Argument	Description
`freq`	The frequency used when including substrate effects during high-frequency extractions Units: Hz Default: 0

Description

The `OPERATING_FREQUENCY` command specifies the frequency used when including substrate effects during high-frequency extractions.

See Also

- TSV
- 3D_IC_FLOATING_SUBSTRATE
- 3D_IC_SUBCKT_FILE
- 3D_IC_TSV_COUPLING_EXTRACTION
- TSV_ CELLS
- Through-Silicon Vias

Feedback

OPERATING_TEMPERATURE

Specifies the operating temperature either in the StarRC command file or in a corners file.

Syntax

```
OPERATING_TEMPERATURE: temperature
```

Arguments

Argument	Description
temperature	Operating temperature in degrees Celsius Default: 25 Valid range: -200 to +300 degrees

Description

The `OPERATING_TEMPERATURE` command can be used in either the StarRC command file or in a corners file.

Use the command as follows:

- In the StarRC command file

  The `OPERATING_TEMPERATURE: temperature` command must be used in the command file if you want to use derating information contained in the nxtgrd file. If the resistance of a layer is changed by the mapping file, the resistance value in the mapping file is used for derating and the ITF value is ignored. The operating temperature is documented in the DESIGN_FLOW TEMPERATURE header in the SPEF file.

- In a corners file

  For simultaneous multicorner extraction, you can set different operating temperatures for different corners by including an `OPERATING_TEMPERATURE: temperature` command in each corner definition in the corners file.

Examples

In this example, rcworst.nxtgrd is extracted at 125° C.

```
OPERATING_TEMPERATURE: 125
TCAD_GRD_FILE: rcworst.nxtgrd
```

Feedback

See Also

- NETLIST_FORMAT
- TCAD_GRD_FILE
- CORNERS_FILE
- SIMULTANEOUS_MULTI_CORNER
- Simultaneous Multicorner Extraction

Feedback

PARASITIC_EXPLORER_ENABLE_ANALYSIS

Specifies whether to set up the GPD for later use with the Parasitic Explorer tool.

Syntax

```
PARASITIC_EXPLORER_ENABLE_ANALYSIS: YES | NO
```

Arguments

Argument	Description
YES	Saves additional data necessary for using the Parasitic Explorer tool
NO (default)	Does not enable later user of the Parasitic Explorer tool

Description

The Parasitic Explorer tool allows you examine the contents of a GPD.

When you create a GPD intended for later use with the Parasitic Explorer tool, you must set the PARASITIC_EXPLORER_ENABLE_ANALYSIS command to YES during the extraction to ensure that the GPD contains the necessary information.

For more information, see the *Parasitic Explorer User Guide*.

See Also

- The Parasitic Explorer Tool for Querying GPD Contents

Feedback

PCELL_EXTRACTION_FILE

Specifies a file for extracting settings of parameterized cells (PCell). Valid only for transistor-level extraction.

Syntax

```
PCELL_EXTRACTION_FILE: file_name
```

Arguments

Argument	Description
file_name	The name of the file in which the following arguments are specified: *cell_name* [KEEP_CC | IGNORE_CC] [KEEP_CG | IGNORE_CG | KEEP_ALL_CG | AUTOMATIC_CG_HANDLING] [SUPERCONDUCTIVE | CONDUCTIVE] For the description of the arguments, see COUPLE_TO_SPECIFIED_PCELL_PINS and EXTRACT_SPECIFIED_PCELL_PINS.

Description

When you use the PCELL_EXTRACTION_FILE command, the command settings overrides and ignores all the settings even if it is specified with the following commands for the specified PCell:

- SKIP_PCELLS
- COUPLE_TO_PCELL_PINS
- COUPLE_TO_SPECIFIED_PCELL_PINS
- EXTRACT_SPECIFIED_PCELL_PINS

The command requires you to specify a cell name and a setting as arguments in a file. The following usage notes apply:

- You can use the * and ? wildcards in the cell name. The cell name is case-sensitive if the CASE_SENSITIVE command is set to YES (the default).
- You can use this command multiple times in a StarRC command file. If specific PCells are named multiple times, later instances of this command overwrite earlier instances.
- All cells named in this command must also be named in a SKIP_PCELLS command. If a cell name does not appear in both commands, the tool issues a warning message and ignores the PCELL_EXTRACTION_FILE command setting for that cell.

Feedback

Using Parameterized Cells With Device Recognition Layer

When you list the names of PCells with the following commands, the name of the PCell and device recognition layer combination might not always be unique. During these cases, the StarRC tool allows selection of PCells based on unique combinations.

- `PCELL_EXTRACTION_FILE`
- `SKIP_PCELLS`
- `COUPLE_TO_PCELL_PINS`
- `COUPLE_TO_SPECIFIED_PCELL_PINS`
- `EXTRACT_SPECIFIED_PCELL_PINS`

To specify device recognition layer from the IC Validator runset or Calibre rule file, use the `pcell_name_wildcard [ "[" seed_layer_name "]" ]` argument syntax with the commands to select a more specific PCells as shown in the following example:

```
SKIP_PCELLS: my_pcell* [my_reg]
```

The StarRC tools considers only the PCells whose names begin with `my_pcell` and also with a device recognition layer `my_reg`.

See Also

- SKIP_PCELLS
- COUPLE_TO_PCELL_PINS
- COUPLE_TO_SPECIFIED_PCELL_PINS
- EXTRACT_SPECIFIED_PCELL_PINS
- Preparing the Mapping File

Feedback

PIN_CUT_THRESHOLD

Takes a long port and splits it into multiple *|P nodes.

Syntax

```
PIN_CUT_THRESHOLD: distance_in_nm
```

Arguments

Argument	Description
distance_in_nm	Distance at which to cut a long port Units: nm Default: none

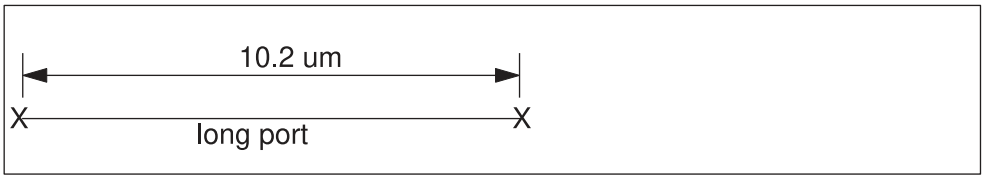
Description

The PIN_CUT_THRESHOLD command splits a long port into multiple *|P nodes. If a port has a continuous Manhattan length less than the PIN_CUT_THRESHOLD, the StarRC tool creates one *|P node for that segment. If a long port is not evenly divisible by the PIN_CUT_THRESHOLD value, the tool breaks the port into symmetric segments, as shown in Figure 216

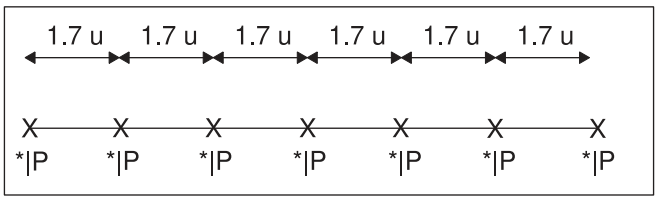
When you use the PIN_CUT_THRESHOLD command, the StarRC tool internally sets the short_pins command to NO.

This command supports only Manhattan port shapes and LEF/DEF, Milkyway, IC Validator, and Hercules flows. It does not support the Calibre Connectivity Interface flow.

Figure 216 Symmetric Division



With a PIN_CUT_THRESHOLD value of 2,000 nm, this 10.2 um long port would be cut as follows:



The output in the SPEF netlist would be as follows:

Examples

```
PIN_CUT_THRESHOLD: 500

…
*CONN
*P NET1_x40000y105000 B *C 40.00 105.00
*P NET1_x45000y115500 B *C 45.00 115.50
*P NET1_x50000y115500 B *C 50.00 115.50
*P NET1_x55000y115500 B *C 55.00 115.50
….
*I Level_1/NET_A:PIN_A_x10000y10000 *C 10.00 10.00
*I Level_1/NET_A:PIN_A_x15000y20500 *C 15.00 20.50
*I Level_1/NET_A:PIN_A_x20000y20500 *C 20.00 20.50
*I Level_1/NET_A:PIN_A_x25000y20500 *C 25.00 20.50
```

See Also

- SHORT_PINS
- INSTANCE_PORT

Feedback

PIO_FILE

Specifies a file containing primary pin direction and loading capacitance.

Syntax

```
PIO_FILE: file1 file2  … fileN
```

Arguments

Argument	Description
fileN	File containing the pin descriptions Default: none

Description

This command specifies a file containing primary pin direction and loading capacitance. Only applicable for top-level ports. Information contained in `PIO_FILE` is applied to the output netlist. Format is white-space-delimited with one entry per line:

```
pin_name IN | OUT | BIDIR [cap cap_value]
```

Examples

```
IN1 IN
CLK IN
OUT OUT cap 5pf
IN2 IN
OUT2 OUT cap 1e-12
```

See Also

- NETLIST_FORMAT

Feedback

PLACEMENT_INFO_FILE

Specifies the generation of output placement information.

Syntax

```
PLACEMENT_INFO_FILE: YES | NO
```

Arguments

Argument	Description
YES	Creates a placement information file
NO (default)	Does not provide placement information

Description

DSPF netlists can be either flat or hierarchical. In a flat extraction, all nodes are shown with respect to origin of the top cell. However, in a hierarchical flow, each node in a hierarchical cell's DSPF is shown with respect to its origin. To map these nodes to the next level of hierarchy, a downstream analysis tool must know the placement of the cell in the next level of hierarchy with rotation and flip attributes.

When you set the PLACEMENT_INFO_FILE command to YES, the StarRC tool creates a file whose name is specified by the PLACEMENT_INFO_FILE_NAME command that contains the following information:

- Angle
- Reflection
- Location of the cell (x- and y-coordinates)
- Cell name
- Cross-referenced instance name

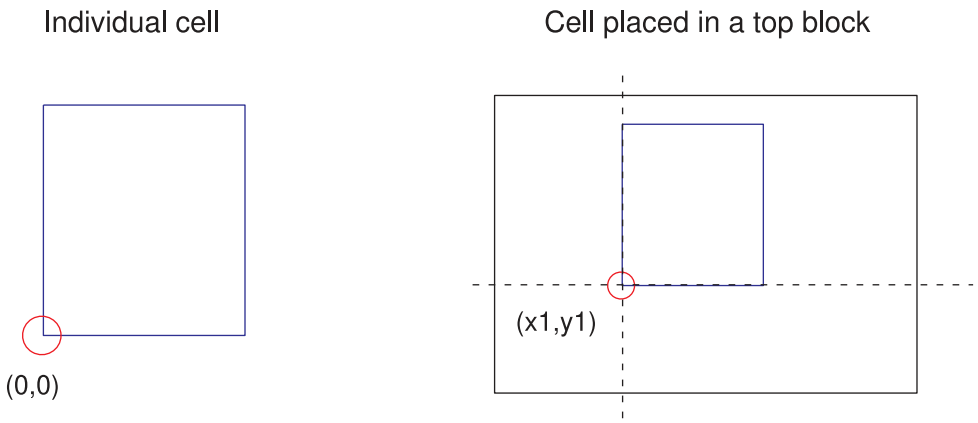
The cell location written to the file is relative to the cell origin unless the cell is instantiated in a top design block, in which case the location is relative to the top block origin.

The number of digits after the decimal point for the x- and y-coordinates is based on the precision of the input design files, as follows:

- If the input design precision is 1 nm, the number of digits after the decimal point is 3.
- If the input design precision is 0.1 nm or better (in other words, a smaller number), the number of digits after the decimal point is 4.

Feedback

Figure 217 Cell Location



Examples

The following is an example of a transistor-level placement file.

```
* PLACEMENT FILE
* VENDOR: "Synopsys Inc."
* PROGRAM: "StarRC"
...

TOP_CELL = STBASE
inst_name cell_name X_Coord Y_Coord Angle reflection
xSD_8/M1 P 8.550 54.010 0 NO
xSD_8/M2 N 8.550 17.225 0 NO
xSD_9/M1 P 8.550 56.812 0 NO
xSD_9/M2 N 8.550 18.100 0 NO
xSD_10/M1 P 8.550 58.800 0 NO
xSD_10/M2 N 8.550 20.500 0 NO
xSD_11/M1 P 15.975 54.010 0 NO
xSD_11/M2 P 8.550 50.225 0 NO
xSD_11/M3 P 29.117 60.459 0 NO
xSD_11/M4 N 29.117 21.100 0 NO
xSD_11/M5 N 8.550 25.500 0 NO
xSD_11/M6 N 15.975 23.300 0 NO
* End of File
```

This is an output example from a placement file. A transistor-level placement file example follows the argument list.

```
* PLACEMENT FILE
* VENDOR "Synopsys, Inc."
* PROGRAM: "StarRC"
...

TOP_CELL = SMALLARRAY
```

```
inst_name cell_name X_Coord Y_Coord Angle reflection
xSD_0 STBASE -1925.825 -379.700 0 NO
xSD_1 STBASE -1797.200 -379.700 0 NO
xSD_2 STBASE -1668.650 -379.700 0 NO
xSD_3 STBASE -1540.025 -379.700 0 NO
xSD_4 STBASE -1411.575 -379.700 0 NO
xSD_5 STBASE -1282.875 -379.700 0 NO
xSD_6 STBASE -1154.200 -379.700 0 NO
xSD_7 STBASE -1025.650 -379.700 0 NO
```

See Also

- PLACEMENT_INFO_FILE_NAME

Feedback

PLACEMENT_INFO_FILE_NAME

Specifies the name of the placement info file.

Syntax

```
PLACEMENT_INFO_FILE_NAME: info_file
```

Arguments

Argument	Description
info_file	The generated placement information file name Default: none

Description

Use this command to specify a file name when you create a placement info file by setting the `PLACEMENT_INFO_FILE` command to `YES`.

Examples

```
PLACEMENT_INFO_FILE_NAME: top_block.place
```

See Also

- PLACEMENT_INFO_FILE

Feedback

POWER_EXTRACT

Specifies the extraction of power nets.

Syntax

```
POWER_EXTRACT: YES | NO | RONLY | DEVICE_LAYERS [NON_DEVICE_LAYERS_CONLY]
```

Arguments

Argument	Description
YES	Extracts the power nets
NO (default)	Does not extract the power nets
RONLY	Extracts the power nets for resistance only. Creates an additional resistor-only netlist when the NETLIST_POWER_FILE command is also used.
DEVICE_LAYERS	Extracts resistance and capacitance for the power nets whose layers are specified with the device_layer keyword in a conducting_layers statement in the mapping file. Valid only for transistor-level flows (Hercules, IC Validator, and Calibre flows).
DEVICE_LAYERS NON_DEVICE_LAYERS _CONLY	Extracts capacitance for the power nets on all layers. Extracts the resistance for the power nets whose layers are specified with the device_layer keyword in a conducting_layers statement in the mapping file. Valid only for transistor-level flows (Hercules, IC Validator, and Calibre flows).

Description

By default, the StarRC tool does not extract power nets. Coupling capacitance between power nets and signal nets is taken into consideration when the signal nets are extracted. However, the power nets are not included in the netlist and the resulting effect on the signal nets is reported as a grounded capacitance. You can extract power nets by setting the POWER_EXTRACT command to a setting other than NO.

The power nets are identified as follows:

- Power nets are identified implicitly in a routed NDM format or Milkyway design database or a LEF/DEF layout description.
- For transistor-level flows, you can specify the power nets to be extracted by using the POWER_NETS command. If the POWER_NETS command is not used, the power net definition is inherited from the LVS tool.

Feedback

The coupling capacitances between the signal and power nets are kept and grounded. The coupling capacitances between power nets are not extracted, and the total capacitance reported for power nets is zero.

You can control the amount of reduction for power nets by using the `POWER_REDUCTION` command.

Special-purpose options are available, as follows:

- The `RONLY` option

  The `POWER_EXTRACT:RONLY` command creates an additional resistor-only netlist when the `NETLIST_POWER_FILE` command is also used. The signal netlist (the standard netlist) contains resistance and capacitance parasitics of the signal nets. The power netlist contains resistance parasitics of the power nets.

- The `DEVICE_LAYERS` option (transistor-level flows only)

  The StarRC tool extracts resistance and capacitance from power nets in layers that contain the `device_layer` keyword within a `conducting_layers` or `via_layers` statement in the mapping file.

  This option specifies a special flow in which layers on power nets other than device layers are treated as superconductive layers. During extraction, the StarRC tool connects nodes from superconductor layers to top-level power ports by using shorting resistors. The result in the netlist is one *|P entry that connects to millions of device-level interaction nodes.

  To simplify processing and reduce the number of resistors on a node, the tool connects the superconductive nodes within a partition, then makes a single connection to a top-level port. You cannot control how the superconductive nodes and ports are connected.

  Note the following:

  - If multiple isolated power straps connect to a top-level port, the node processing procedures might cause the isolated regions to be connected.
  - If physical opens exist on power nets in non-device layers, the StarRC tool does not recognize them because the node processing procedures mark these layers with the same net ID. As a result, the opens reported for the `DEVICE_LAYERS` option might be different from the opens reported for the `YES` option.

- The `DEVICE_LAYERS NON_DEVICE_LAYERS_CONLY` option (transistor-level flows only)

  The StarRC tool extracts capacitance from power nets on all layers and resistance from power nets in layers that contain the `device_layer` keyword within a `conducting_layers` or `via_layers` statement in the mapping file.

Feedback

Examples

In the following example, the StarRC tool extracts contact, gate, and diffusion resistance for VDD and VSS power nets based on the `device_layer` keyword in the mapping file. The instance section netlist lists all the devices connected to these power nets along with the signal nets.

```
NETS:  (any definition)
POWER_NETS: VDD VSS
POWER_EXTRACT: DEVICE_LAYERS
```

See Also

- NETLIST_POWER_FILE
- NETS
- POWER_NETS
- ENHANCED_GPD_POWER_REDUCTION
- POWER_REDUCTION
- conducting_layers
- via_layers

Feedback

POWER_NETS

Specifies power nets for special treatment during extraction.

Syntax

```
POWER_NETS: netnames
```

Arguments

Argument	Description
netnames	List of power net names, delimited by white space . Default: none

Description

The StarRC tool obtains the list of power nets from different sources, depending on the design database and the setting of the POWER_NETS command, as described in Table 84.

The *, ?, and ! wildcards are accepted. Case sensitivity is determined by the CASE_SENSITIVE command.

Table 84 Power Net Identification

Database	With POWER_NETS command	Without POWER_NETS command
Fusion Compiler, IC Compiler II, Milkyway, and LEF/DEF	The power nets implicitly defined in the database plus the nets specified in the POWER_NETS command	The power nets implicitly defined in the database
Hercules	The POWER_NETS command alone (the Hercules runset is ignored)	Power and ground net definition from the Hercules runset
IC Validator (Milkyway XTR view flow)	The POWER_NETS command alone (the IC Validator runset report file is ignored)	Power and ground net definition from the IC Validator runset report file
IC Validator (annotated GDS (AGDS) flow)	A combination of LVS power and ground names are used from the ICV Runset report file and the POWER_NETS command settings	Power and ground net definition from the IC Validator runset report file
Calibre Connectivity Interface (CCI)	The CCI LVS extraction report file plus the nets specified in the POWER_NETS command	Power and ground net definition from the CCI LVS extraction report file

Feedback

See Also

- POWER_EXTRACT
- The Power Nets Report

Feedback

POWER_PORTS

Specifies a list of patterns for identifying power and ground ports for skip cells if their types are not explicitly defined in the design database and their names are different from the top-level nets specified in the POWER_NETS command.

Syntax

```
POWER_PORTS: wildcard_list
```

Arguments

Argument	Description
wildcard_list	List of port names to identify to the power nets Default: List specified by the POWER_NETS statement

Description

For LEF/DEF designs, power ports are specified by the USE POWER and USE GROUND keywords in LEF MACRO PIN definitions.

For Milkyway designs, the port-type tables defined at stream-in and during BLOCKAGE_PIN_VIA_EXTRACTION determine the usage. Querying the PIN shapes in the FRAM views indicates their usage type. For Hercules or IC Validator XTR view input, none of the ports are marked.

By default, the POWER_PORTS command inherits the POWER_NETS list.

See Also

- NETLIST_POWER_FILE
- POWER_NETS

Feedback

POWER_REDUCTION

Specifies the amount and type of reduction to perform for extracted power nets. Valid only for transistor-level flows.

Syntax

```
POWER_REDUCTION: HIGH | YES | NO | LAYER | LAYER_NO_EXTRA_LOOPS
```

Arguments

Argument	Description
HIGH	Performs maximum netlist reduction on power nets
YES (default)	Performs standard netlist reduction on power nets
NO	Does not perform netlist reduction on power nets
LAYER	Performs standard netlist reduction on power nets on the same layer only. Reduction is not applied across different conductor layers.
LAYER_NO_EXTRA_LOOPS	Performs standard netlist reduction on power nets on the same layer and ensures that no resistive loops are introduced. Loops that result from the layout topology are not removed, such as meshes that result from overlapping metals connected by parallel vias.

Description

If power nets are extracted, the POWER_REDUCTION command allows you to specify the amount of parasitic reduction for power nets separately from signal nets. Power nets are typically very large and usually have a smaller effect on circuit behavior than signal nets. Therefore, performing more reduction on power nets than on signal nets can reduce the netlist size without sacrificing performance.

By default, power nets are not extracted and the POWER_EXTRACT command is set to NO. The POWER_REDUCTION command has an effect only for POWER_EXTRACT settings other than NO.

If the REDUCTION command is set to HIGH, the POWER_REDUCTION command does not reduce the netlist size further, because both signal nets and power nets are already reduced.

See Also

- POWER_EXTRACT
- REDUCTION

Feedback

PRINT_SILICON_INFO

Prints resistor silicon width or area in the netlist tail comments.

Syntax

```
PRINT_SILICON_INFO: YES | NO
```

Arguments

Argument	Description
YES	Prints silicon width $si_w for conductor resistors and nonphysical resistors
NO (default)	Does not print silicon width

Description

Downstream simulation tools sometimes require the resistor silicon width value, which is different from the drawn width. The PRINT_SILICON_INFO command prints the silicon width in addition to the drawn width. The extra information is printed in the netlist tail comments, therefore the NETLIST_TAIL_COMMENTS command must also be set to YES.

The StarRC tool prints the silicon width $si_w for conductor resistors and nonphysical resistors in addition to the existing length, width, and process layer values ($l, $w and $lvl). For via resistors, the silicon area $si_a is provided after the existing area and process layer values ($a and $lvl).

The silicon width might be smaller or larger than the drawn width. ITF commands such as the ETCH and ETCH_VS_WIDTH_AND_SPACING commands affect the silicon width computation. The silicon width is the value used in resistance calculations.

The silicon area $si_a for a via resistor is always the same as the area $a because via etch is not supported in resistance calculation.

Examples

The following portion of a netlist shows the silicon width value at the end of the tail comments for conductor resistor R41 and the silicon area value at the end of the tail comments for via resistor R42:

```
R41 M15:SRC Y:15 11.6946 $l=0.105 $w=0.153 $lvl=100 $si_w=0.149
R42 Y:12 Y:54 30 $a=0.0081 $lvl=134 $si_a=0.0081
```

Feedback

The simultaneous multicorner flow can generate either individual netlists for each corner or a single merged netlist. The silicon width is reported in the SMC flow as follows:

- For an individual netlist, the netlist tail comments values come from that single corner.
- For a standard merged netlist, the netlist tail comments values come from the primary corner, which is the first corner in the colon-delimited list in the `SELECTED_CORNERS` command.

See Also

- NETLIST_TAIL_COMMENTS

Feedback

PRINT_FSCOMPARE_REPORT

Prints comparison results between StarRC and field solver extraction for specified nets.

Syntax

```
PRINT_FSCOMPARE_REPORT: YES | NO
```

Arguments

Argument	Description
YES	Generates comparison reports
NO (default)	Does not print any report

Description

The command prints total capacitance and coupling capacitance reports. The total capacitance report is suffixed with “comptot”, and the coupling capacitance report is suffixed with “compcoup”.

To generate the reports, you should use the `PRINT_FSCOMPARE_REPORT: YES` command with the `FS_EXTRACT_NETS` command. Nets to be compared are specified with the `FS_EXTRACT_NETS` command. You can specify the nets using wildcards. The generated report compares results for all the nets that match with the wildcards setting.

Note that the StarRC tool always generates reports when you use the `EXTRACTION: FSCOMPARE` command even if the `PRINT_FSCOMPARE_REPORT` command is not set to `yes`.

See Also

- FSCOMPARE Flow Output Files
- NETLIST_TAIL_COMMENTS

Feedback

PRINT_VIA_VARIATION_MODEL

Prints via variation model names.

Syntax

```
PRINT_VIA_VARIATION_MODEL: YES | NO
```

Arguments

Argument	Description
YES	Prints via variation model names
NO (default)	Does not print variation model names even if the names are specified in the ITF file

Description

Prints via variation model names to netlist files so that the PrimeTime tool can apply via variation models based from the netlist file.

If you use the `NETLIST_TAIL_COMMENTS: NO` and `PRINT_VIA_VARIATION_MODEL: YES` commands, the StarRC tool issues a warning message and sets the `PRINT_VIA_VARIATION_MODEL` command to `NO`. However, when you use the `NETLIST_TAIL_COMMENTS: YES` and `PRINT_VIA_VARIATION_MODEL: YES` commands, the tool prints the following information:

- Via variation model map in the netlist file header.
- The `$vc_model=`*`model_name_id`* string in the netlist tail comments of the via resistor section.

In the Simultaneous multicorner (SMC) flow, the StarRC tool checks the consistency of via variation models among corners and errors out if there is any difference.

Examples

```
* VIA_MODEL_NAME_MAP
*1 vc_m_1a
*2 vc_m_1b
*3 vc_m_2

…

R9_7 clk:4 clk:5 98.383781 $vc=12 $savl=2 $savc=2 $cover_dir=0
 $a=0.00028000
$lvl=380 $llx=0.0930 $lly=0.0620 $urx=0.1070 $ury=0.0820  $vc_model=2
```

See Also

- VIA_COVERAGE
- SIMULTANEOUS_MULTI_CORNER
- Simultaneous Multicorner Extraction
- Via Coverage
- NETLIST_TAIL_COMMENTS

Feedback

PROBE_TEXT_FILE

Specifies a file that contains simulation probe points to appear in the parasitic netlist.

Syntax

```
PROBE_TEXT_FILE: file_name
```

Arguments

Argument	Description
file_name	The name of a file that contains defined probe points Coordinate units: microns Default: none

Description

A probe point is a specific node point in the parasitic node mesh for a particular net.

The coordinates in the probe text file are in microns. A coordinate is an absolute value and is not related to precision. The HALF_NODE_SCALE_FACTOR and the MAGNIFICATION_FACTOR commands are applied.

A PROBE_TEXT_FILE entry is translated and written to the netlist if the following are true:

- The specified cell reference name is a valid extracted cell. If XREF is in the command file, the cell name must correspond to a valid LVS equivalence point. The CELL_TYPE command regulates whether the schematic or layout name is used.
- A polygon at the specified layer name exists at the specified coordinate location.
- The probe point falls on a net for which parasitics are included in the netlist.
- The probe point falls on a cross-referenced net within a cross-referenced instance in the XREF: COMPLETE command.
- The line is less than 265 characters long.

Feedback

Probe points take the following formats in the output netlist:

- Instance level: **OI

  For probe text defined at the instance level, regardless of whether the probe text overlaps any routing or port polygons

- Top level: **OP

  For probe text defined at the top level, regardless of whether the probe text overlaps any routing or port polygons

Examples

Example 34 shows the syntax of the probe text file.

Example 34 Probe Text File Syntax

```
CELL cell_name
textstring local_x local_y layername
textstring local_x local_y layername
...
```

Parameter	Definition
textstring	Identifying layout text label for the probe point by which the point is identified in the parasitic netlist.
local_x	X-coordinate for the probe point location. The specified coordinate is interpreted local to the specified cell.
local_y	Y-coordinate for the probe point location. The specified coordinate is interpreted local to the specified cell.
layername	Database layer name to which the probe point corresponds. The name must correspond to a layer mapped in the `conducting_layers` section of the mapping file.
cell_name	Name of the cell master in which the probe point is specified. The `CELL_TYPE` command regulates whether a layout cell or schematic cell is used.

An example of a probe text file is as follows:

```
CELL ADD4_TOP
   PROBE1 10   10   M3
CELL INVX$
   GATE_1 16.3 14.5 GPOLY
   GATE_2 15.3 1.1  GPOLY
```

An example of observation points in a netlist is as follows:

```
**|OP (PROBE1 Z 0 10 10)
**|OI (I1:GATE_1 I1 GATE_1 Z 0 26.3 19.5)
**|OI (I1:GATE_2 I1 GATE_2 Z 0 25.3 6.1)
```

See Also

- CELL_TYPE

Feedback

QTF_MAPPING_FILE

Specifies the location of QuickCap mapping file. Used only in a QTF flow.

Syntax

```
QTF_MAPPING_FILE: qtf_file_location
```

Arguments

Argument	Description
qtf_file_location	Location of the QTF mapping file Default: none

Description

The `QTF_MAPPING_FILE` command specifies the location of a QuickCap mapping file, which is used during mapping when StarRC extraction uses the QTF file. QTF files are always processed by the StarRC field solver. Runtime for the QTF flow might be longer than for standard extraction flows.

The StarRC tool automatically generates the QTF mapping file only if you have specified the `QTF_MAPPING_FILE`, `FS_QTF_FILE` and `MAPPING_FILE` commands in the StarRC command file.

When the `QTF_MAPPING_FILE` command is specified in the StarRC command file and if the `map_qtf_layers` and `qtf_layers` sections

- Exist in the mapping file: The `QTF_MAPPING_FILE` command generates the new layer mapping information to update in the `map_qtf_layers` and `qtf_layers` sections in the mapping file and issues the following warning message

  ```
  (Warning) map_qtf_layers and qtf_layers sections in MAPPING_FILE will
   be overwritten (SX-3911)
  ```

- Do not exist in the mapping file: The `QTF_MAPPING_FILE` command generates both the `map_qtf_layers` and `qtf_layers` sections and issues the following information message

  ```
  (Information) map_qtf_layers and qtf_layers sections will be generated
   from QTF_MAPPING_FILE (SX-3910)
  ```

See Also

- MAPPING_FILE
- FS_QTF_FILE

Feedback

- map_qtf_layers
- qtf_layers
- The QTF Flow

Feedback

RC_SCALING_FILE

Specifies a scaling file that contains scaling factors to scale capacitance and resistance on conducting layers and resistance on via layers.

Syntax

```
RC_SCALING_FILE: file_name
```

Arguments

Argument	Description
file_name	The name of a file with scaling factors

Description

This command specifies scaling factors to scale capacitance and resistance on the following layers:

- Conducting layers: Use the following factors to scale capacitance and resistance:
  - Coupling capacitance (CC) on the same layers with a constant factor and a factor table, based on the width and spacing value of the self and neighbor conducting layers
  - Coupling capacitance across layers with a constant factor
  - Total capacitance (CT) with a constant factor
  - Resistance with a constant factor and a factor table based on the self-width and the spacing value with neighbors on both sides
  - Thickness with a constant factor (see Setting Thickness for Condutor Layer)
- Via layers: You can scale resistance with a constant factor.

The mask color is also supported for coupling and resistance. For information, see Mask Color Settings.

When you specify the `RC_SCALING_FILE` statement with the `SIMULTANEOUS_MULTI_CORNER` command, the same scaling factors are used for different corners. For information about enabling different corners simultaneously, see `SIMULTANEOUS_MULTI_CORNER`.

The syntax of `RC_SCALING_FILE` is as follows:

```
CONDUCTOR itf_layer_name {
   CC_SCALING_FACTOR {
```

Feedback

```
    [NUMBER_OF_MASKS = number_of_masks
    MASKS(a,b) [(c,d)…] {]
      SPACINGS (spacing) {
         WIDTHS1 {w1, w2, … wn}
         WIDTHS2 {w1, w2, … wn}
         VALUES {v(w1, w1) v(w2, w1) … v(wn, w1)
                 v(w1, w2) v(w2, w2) … v(wn, w2)
                 …
                 v(w1, wn) v(w2, wn) … v(wn, wn)}
      }
      SPACINGS (spacing) {
         …
      }
     …
    }
    [MASKS(i,j) [(k,l)] {
      …
     }]
  }
  R_SCALING_FACTOR {
    [NUMBER_OF_MASKS = number_of_masks
    MASKS(a) [(b)…] {]
      WIDTHS (width) {
         SPACINGS1 {s1, s2, … sn}
         SPACINGS2 {s1, s2, … sn}
         VALUES {v(s1, s1) v(s2, s1) … v(sn, s1)
                 v(s1, s2) v(s2, s2) … v(sn, s2)
                 …
                 v(s1, sn) v(s2, sn) … v(sn, sn)}
      }
      WIDTHS (width) {
         …
      }
      …
     }
    …
  }
  CC_SCALING_FACTOR = value
  R_SCALING_FACTOR = value
  CT_SCALING_FACTOR = value
  CROSS_LAYER_CC_SCALING_FACTOR {
     (cross layer ITF name, value)
     ……
     (cross layer ITF name, value)
  }
  THICKNESS_SCALING_FACTOR = value
}
VIA itf_layer_name {
  R_SCALING_FACTOR = value
}
```

Feedback

Scaling Factor Rules

The RC_SCALING_FILE command usage requirements are as follows:

- You need to specify an ITF layer name after the CONDUCTOR or VIA keyword. You can specify each layer only one time. The StarRC tool issues an error message if the layer name is not found in the nxtgrd file.
- You need to specify each type of scaling factor only one time on each layer. The StarRC tool issues an error message if you specify the same type of scaling factor more than one time on a single layer.
- The CC_SCALING_FACTOR and R_SCALING_FACTOR scaling factors have the following forms:
  - A constant value
  - A factor table

  The constant value and the factor table can be specified together on the same layer. When both forms are specified on the same layer, both scaling factors are applied. Example 35 shows that coupling capacitances are specified on the M2 layer. The resistances are specified on the M2 layer with R_SCALING_FACTOR similar to the way as shown in Example 35.

Example 35 Specifying Coupling Capacitances on M2 Layer

```
CONDUCTOR M2 {
   CC_SCALING_FACTOR = 0.6
   CC_SCALING_FACTOR {
      SPACING (0.04) {
      WIDTHS1 {0.04 0.06}
      WIDTHS2 {0.04 0.06}
      VALUES {0.7 0.8 0.8 0.9}
      }
   }
}
```

In Example 35, coupling capacitances on the M2 layer are scaled by the constant value, as well as the width and spacing dependent factor table. The 0.6 is the constant value. The 0.7 value is used from the factor table if the widths and spacing values are 0.04. The following equation shows how coupling capacitances are scaled:

```
CC' = CC * 0.7 * 0.6
```

- CT_SCALING_FACTOR scales all coupling and ground capacitances related to the specified layer.

Table 85 lists the rules for scaling factor table.

Feedback

Table 85 Rules for Scaling Factor Table

Scaling factors	Rules description
CC_SCALING_FACTOR and R_SCALING_FACTOR	• Width and spacing values must be in increasing order. • Width and spacing values are from the original dimension. • When the width value is not at any index, the maximum width index which is smaller than the width value is applied. • When the spacing value is not at any index, linear interpolation is applied. • When the width or spacing value is outside the table range, the boundary value is applied.
CC_SCALING_FACTOR	The 2-D table for each spacing must be symmetric. Otherwise, the StarRC tool issues an error message.
R_SCALING_FACTOR	• The 2-D table for each width must be symmetric. Otherwise, the StarRC tool issues an error message. • When the two spacing values are the same, linear interpolation is applied on a diagonal line, instead of two directions. For example, if the two spacings values are between s1 and s2, linear interpolation is done between v(s1,s1) and v(s2,s2), instead of v(s1,s1), v(s1,s2), v(s2,s1), and v(s2,s2).
CROSS_LAYER_CC_SCALING_FACTOR	• The cross layer name cannot have the same name as the conducting layer name. Otherwise, the StarRC tool issues an error message. • The same scaling factors must be specified on both conducting layers.

Mask Color Settings

You can specify the mask color settings in RC_SCALING_FILE with the following keywords:

- NUMBER_OF_MASKS: Specify a value to indicate the number of masks in a layer. When you use the NUMBER_OF_MASKS keyword, at least one colored mask combination must be specified with the MASKS keyword.
- MASKS: Specify one or more mask pair for the CC_SCALING_FACTOR table and one or more masks for the R_SCALING_FACTOR table.

  In the CC_SCALING_FACTOR table, the MASKS keyword format is MASKS(mask1, mask2). Where, MASKS(1,2) and MASKS(2,1) are equivalent. You can specify the same mask pair only one time in the same table for the same layer.

  In the R_SCALING_FACTOR table, the MASKS keyword format is MASKS(mask). You can specify the same mask only one time in the same table for the same layer.

Feedback

Table 86 shows how the tool scales the color scaling factor based on the mask color settings in the design and in RC_SCALING_FILE. Also,

- For any mask pairs or masks do not require scaling, these masks need not be specified and the StarRC tool sets the scaling factor to 1.
- 0 indicates a colorless mask. If you do not specify a value for colorless layers, the scaling factor is set to 1.
- If you do not use the NUMBER_OF_MASKS keyword, the scaling factor is applied to all colored layers if any.

Table 86 Scaling Behavior Based on Color Settings in Design and RC_SCALING_FILE

Color in a design	Color in RC_SCALING_FILE	Scaling behavior
Y	N	Scales all colored layers.
Y	Y	Scales based on the RC_SCALING_FILE table.
N	N	Uses the colorless table .
N	Y	Uses only non-color table (mask = 0). Ignores any table with mask > 0.

Setting Thickness for Condutor Layer

You can specify the thickness for each conductor layer in RC_SCALING_FILE with the THICKNESS_SCALING_FACTOR keyword. Also, there can be only one instance of THICKNESS_SCALING_FACTOR for each layer. This setting is valid in both gate-level and transistor-level flows and in advanced technology nodes (based on the POLYNOMIAL_BASED_THICKNESS_VARIATION table) .

Example 36 shows how to specify the thickness scaling factor for the conductor M2 layer.

Example 36 Specifying Thickness on M2 Layer

```
CONDUCTOR M2 {
   THICKNESS_SCALING_FACTOR = 0.8
   CC_SCALING_FACTOR = 0.6
   CC_SCALING_FACTOR {
      SPACING (0.04) {
      WIDTHS1 {0.04 0.06}
      WIDTHS2 {0.04 0.06}
      VALUES {0.7 0.8 0.8 0.9}
      }
   }
}
```

The tool applies the thickness scaling factor to the final thickness only after applying the top and bottom thickness to the conductor layer. The change in the final thickness is calculated using the following formula:

```
thickness_scaled = thickness_layer * (change+1) *
 thickness_scaling_factor
```

See Also

- SIMULTANEOUS_MULTI_CORNER

Feedback

REDUCTION

Specifies parasitic netlist reduction options for signal nets.

Syntax

```
REDUCTION: HIGH | YES | NO | LAYER | NO_EXTRA_LOOPS |
 LAYER_NO_EXTRA_LOOPS | ADVANCED
```

Arguments

Argument	Description
HIGH	Performs maximum reduction; the netlist is smaller than with the YES setting
YES (default)	Performs standard reduction
NO	Performs minimal reduction, including removing shorting resistors (if possible) and reducing parallel resistances
LAYER	Similar to YES, but performs reduction on the same layer only, which prevents reducing via nodes
NO_EXTRA_LOOPS	Performs standard reduction and ensures that no resistive loops are introduced, for use with delay calculators that cannot interpret resistive loops. The netlist is 10 to 20 percent larger than with the YES setting. Loops in the parasitic netlist that result from the layout topology are not removed. For example, overlapping metals connected by parallel vias can produce meshes in the parasitic netlist even with this option.
LAYER_NO_EXTRA_LOOPS	Similar to NO_EXTRA_LOOPS, but performs reduction on the same layer only
ADVANCED	Similar to HIGH, but allows using of additional standalone reducer commands (Using the REDUCTION_NETS Command).

Description

This command enables the reduction of extracted parasitic devices for signal nets. Providing a reduced netlist might improve the runtime of downstream simulation tools. The degree of reduction is controlled by the REDUCTION_MAX_DELAY_ERROR command. The reduction algorithm preserves point-to-point resistance and total net capacitance.

To specify reduction for power nets, you must enable extraction of power nets by using the POWER_EXTRACT command and specify the type of reduction by using the POWER_REDUCTION command.

If you use the PrimeTime tool for timing analysis, use the NO_EXTRA_LOOPS setting to simplify the resistor network.

To prevent specific nets from being reduced, use the `INDESIGN_OPEN_NETS` command.

The StarRC tool also provides a standalone reducer that operates on SPEF and DSPF files. For more information, see Standalone Reducer.

The `REDUCTION: ADVANCED` command supports all the optional keywords listed in Using the REDUCTION_NETS Command, except the following keywords:

- `SHORT_PINS`
- `SHORT_RES`
- `SHORT_RES_LAYER`
- `IDEAL, RONLY`
- `RONLY_KEEPC`

See Also

- INDESIGN_OPEN_NETS
- POWER_EXTRACT
- POWER_REDUCTION
- REDUCTION_MAX_DELAY_ERROR
- Standalone Reducer

Feedback

REDUCTION_MAX_DELAY_ERROR

Specifies the acceptable net delay error tolerance during reduction.

Syntax

```
REDUCTION_MAX_DELAY_ERROR: threshold
```

Arguments

Argument	Description
threshold	The absolute delay error Units: seconds Default: 1.0 e-12

Description

The `REDUCTION_MAX_DELAY_ERROR` command controls the parasitic reduction operation by specifying the maximum amount of delay error allowed.

The difference in the absolute delay between the original and reduced netlists cannot be greater than the specified threshold.

See Also

- REDUCTION
- INDESIGN_OPEN_NETS

Feedback

REFERENCE_DIRECTION

Specifies the reference direction of the process technology.

Syntax

```
REFERENCE_DIRECTION: VERTICAL | HORIZONTAL  | NONE
```

Arguments

Argument	Description
VERTICAL	Reference direction is vertical
HORIZONTAL	Reference direction is horizontal
NONE (default)	Reference direction is taken from the ITF file

Description

The REFERENCE_DIRECTION command defines the reference direction for the application of orientation-dependent etch defined by the ETCH_VS_WIDTH_AND_SPACING statement in the ITF file.

The ITF file must contain a reference direction specification. If the reference direction is also specified in the StarRC command file, the setting in the command file takes precedence.

Examples

The following example specifies that the reference direction is horizontal:

```
REFERENCE_DIRECTION: HORIZONTAL
```

See Also

- ETCH_VS_WIDTH_AND_SPACING
- REFERENCE_DIRECTION (ITF command)

Feedback

REMOVE_DANGLING_NETS

Specifies whether to identify dangling nets and reassign them to ground (effectively removing them).

Syntax

```
REMOVE_DANGLING_NETS: YES | NO
```

Arguments

Argument	Description
YES	Specifies that the netlist should not contain dangling nets
NO (default)	Specifies that the netlist should retain dangling nets

Description

Dangling nets are defined as:

- Unrouted database nets (for Milkyway, LEF/DEF, and Calibre Connectivity Interface)
- Nets that have only one connection (for Milkyway, LEF/DEF, Calibre Connectivity Interface, Hercules, and IC Validator flows).

Nets that are connected to a pin port (*|P) are not eligible for removal. In Hercules flows, ports must be generated during Hercules device connectivity extraction using the CREATE_PORTS runset command. Otherwise, the StarRC tool does not consider the port connection and the net is removed. Using the REMOVE_DANGLING_NETS command does not remove layout material; the objects are assigned to ground.

See Also

- REMOVE_FLOATING_NETS

Feedback

REMOVE_DIFFUSION_GATE_OVERLAP

Specifies the removal of the gate-diffusion overlap.

Syntax

```
REMOVE_DIFFUSION_GATE_OVERLAP: YES | NO
```

Arguments

Argument	Description
YES	Removes the gate-diffusion overlap by aligning the diffusion with the gate edges
NO (default)	Keeps the gate-diffusion overlap unchanged. This option is only effective for a trench contact process with raised source and drain etch.

Description

In an actual device, the gate polysilicon might overlap with the diffusion layer, and the real diffusion layer is round or diamond-shaped in the corner next to the gate. Process modeling uses a simple effective profile for the device region where the edge of the diffusion shape is exactly aligned with the gate polysilicon shape.

The StarRC field solver implements this model for better accuracy of the capacitance between the sides and top of the gate to the diffusion (Cfi) and the total gate-to-diffusion capacitance (Cf) with the foundry reference data. The REMOVE_DIFFUSION_GATE_OVERLAP command specifies whether the gate-diffusion overlap is removed.

Feedback

REMOVE_FLOATING_NETS

Specifies whether to removes nets that have no connection by grounding them.

Syntax

```
REMOVE_FLOATING_NETS: YES | NO
```

Arguments

Argument	Description
YES	Specifies that the output netlist should not contain floating nets
NO (default)	Specifies that the output netlist should retain floating nets

Description

When the REMOVE_FLOATING_NETS command is set to YES, the StarRC tool treats the floating nets as noncritical material. In other words, the tool finds the coupling capacitance from signal nets to the noncritical material and then decouples these capacitors. The decoupled capacitance is then added to the ground capacitance of the signal net. The total capacitance of the signal nets accounts for the effects of coupling to floating nets. The floating nets are not shown in the parasitic netlist.

When the command is set to YES, the tool does not completely disregard polygons on floating nets. The goal is to eliminate floating nets from the parasitic netlist while accounting for effects these nets have on real signal nets in the design.

The TRANSLATE_FLOATING_AS_FILL: YES command takes precedence over the REMOVE_FLOATING_NETS: YES command.

See Also

- REMOVE_DANGLING_NETS
- TRANSLATE_FLOATING_AS_FILL

Feedback

REMOVE_FLOATING_PORTS

Grounds floating port nets, eliminating them from capacitance extraction.

Syntax

```
REMOVE_FLOATING_PORTS: YES | NO
```

Arguments

Argument	Description
YES	Grounds floating port nets for capacitance extraction
NO (default)	Includes floating port nets in the parasitic netlist

Description

A floating port is a net with only a top-level connection. By default, the StarRC tool retains the floating ports and their connected nets.

To remove the floating ports from an output file and ground shapes of the floating ports, set the REMOVE_FLOATING_PORTS command to YES.

Feedback

REMOVE_METAL_FILL_OVERLAP

Enables the metal fill reuse flow.

Syntax

```
REMOVE_METAL_FILL_OVERLAP: YES | NO
```

Arguments

Argument	Description
YES	Enables the metal fill reuse flow
NO (default)	Disables the metal fill reuse flow

Description

The REMOVE_METAL_FILL_OVERLAP command allows you to reuse metal fill structures in a timing signoff loop to save overall turnaround time.

When extraction is run with reused metal fill, the StarRC tool resolves shorts between metal fill polygons and signal nets by moving the metal fill structures away from the signal nets. The default spacing for the moved metal fills is the SMIN value of the conductor layer in the ITF file. You can optionally define a different spacing by setting the overlap_fill_spacing value in the conducting_layers statement in the mapping file.

See Also

- conducting_layers
- The Metal Fill Reuse Flow

Feedback

REMOVE_NET_PROPERTY

Specifies a single property name to indicate to the Milkyway layout database which objects should not be extracted as signal nets. Valid only for Milkyway flows.

Syntax

```
REMOVE_NET_PROPERTY: property_name
```

Arguments

Argument	Description
property_name	Specifies that nets with this property are not included in the netlist Default: none

Description

These objects are treated as ground during the extraction, and the influence of these objects is considered when you are extracting the capacitance of neighboring signal nets. See the *Milkyway Environment Data Preparation User Guide* for information about setting object properties in the Milkyway database.

See Also

- MILKYWAY_DATABASE

Feedback

REMOVE_TRIVIAL_INSTANCE_PORTS

Removes skip-cell (black-box cell) ports that are not connected to devices.

Syntax

```
REMOVE_TRIVIAL_INSTANCE_PORTS: NO | YES
```

Arguments

Argument	Description
NO (default)	Does not remove trivial-instance ports
YES	Removes trivial-instance ports

Description

For definition of trivial ports, see EXPLODE_TRIVIAL_INSTANCE_PORTS.

When you set the REMOVE_TRIVIAL_INSTANCE_PORTS command to YES, the StarRC tool removes the trivial-instance ports from the netlist. When the trivial-instance port is removed, the polygons of the port are grounded. The resistance and capacitance are not extracted from the polygons, and the neighbours of the grounded polygons might observe the impact of capacitance.

See Also

- EXPLODE_TRIVIAL_INSTANCE_PORTS
- DELETE_TRIVIAL_INSTANCE_PORTS

Feedback

REMOVE_TRIVIAL_NETS

Removes trivial nets with no connectivity.

Syntax

```
REMOVE_TRIVIAL_NETS: NO | YES
```

Arguments

Argument	Description
NO (default)	Does not remove trivial nets in the netlist
YES	Removes the trivial nets in the netlist with no connectivity

Description

For definition of trivial ports, see EXPLODE_TRIVIAL_INSTANCE_PORTS.

A trivial net is defined as a net consisting of connectivity to only trivial ports. When all trivial ports are removed, the trivial net with no connectivity remains in the netlist unless REMOVE_TRIVIAL_NETS command is set to YES.

See Also

- EXPLODE_TRIVIAL_INSTANCE_PORTS
- DELETE_TRIVIAL_INSTANCE_PORTS
- REMOVE_TRIVIAL_INSTANCE_PORTS
- NET_PREFIX

Feedback

REPORT_METAL_FILL_STATISTICS

Specifies whether to report metal fill statistics.

Syntax

```
REPORT_METAL_FILL_STATISTICS: YES | NO
```

Arguments

Argument	Description
YES	Reports metal fill statistics
NO (default)	Does not report metal fill statistics

Description

The StarRC tool can read metal fill ainformation from a variety of sources, including GDSII files, OASIS® files, and design databases. Advanced summary information is available for metal fill cells read from GDSII or OASIS files. Standard summary information is available for metal fill cells read from all other sources.

If the REPORT_METAL_FILL_STATISTICS command is set to YES, the StarRC tool generates one or more summary reports in the star/summary directory.

Generating Metal Fill Summary Report for Virtual Metal Fill Process

If you have used the VIRTUAL_METAL_FILL_POLYGON_HANDLING command and the REPORT_METAL_FILL_STATISTICS command is set to YES, the tool writes the specified layers and the number of polygons added for each layer in the mf_statistics.sum file, as shown in Example 37. This file is saved in the star/summary directory during the metal fill statistics stage after extraction.

The tool writes the mf_statistics.sum file with the number of virtual metal fill polygons for each layer on the entire design even if the number of metal fill cells are less than 1000 (< 1000) or more than 1000 (> 1000).

For detailed information to generate metal fill summary report and file contents for real metal fill polygons, see Metal Fill Reports.

Example 37 Contents in the mf_statistics.sum File for Virtual Metal Fill

```
<database <mask>  <no. of DB fill  <no. of MFGDS fill  <no. of VMF  <no. of total fill
  layer>             polygons>         polygons>          polygons>      polygons>
   AA       0           0                 0               72367          72367
   U1       0           0                 0              527114         527114
   L2       0           0                 0             2114516        2114516
   P1       0           0                 0             2202255        2202255
   M7       0           0                 0             5527630        5527630
```

```
   I6       0           0                0             6995917        6995917
   I5       0           0                0             7872835        7872835
   I4       0           0                0             4645783        4645783
   M3       0           0                0             8586664        8586664
   M2       0           0                0             3990118        3990118
   M1       0           0                0             1560577        1560577

InitDB         Elp=00:00:00 Cpu=00:00:00 Usr=0.0    Sys=0.0    Mem=370.9
MFStatistics   Elp=00:01:28 Cpu=00:01:22 Usr=79.4   Sys=3.0    Mem=370.9
Done           Elp=00:01:28 Cpu=00:01:22 Usr=79.4   Sys=3.0    Mem=370.9
```

See Also

- METAL_FILL_GDS_FILE
- METAL_FILL_OASIS_FILE
- Metal Fill Reports
- VIRTUAL_METAL_FILL_POLYGON_HANDLING

Feedback

REPORT_SMIN_VIOLATION

Specifies whether to report SMIN violations.

Syntax

```
REPORT_SMIN_VIOLATION: YES | NO
```

Arguments

Argument	Description
YES	Reports SMIN violations
NO (default)	Does not report SMIN violations

Description

To create a report of SMIN violations, set the REPORT_SMIN_VIOLATION command to YES. The tool creates a file named *smin.sum* in the star directory. In addition, the tool writes warning messages in the summary file to indicate that SMIN violations occur and to direct you to see the *smin.sum* file for more information.

An SMIN violation between two neighboring polygons occurs if all of the following conditions are met:

- The drawn distance between the polygons is less than the SMIN value.
- The polygons belong to different nets.
- The polygons are not floating fill polygons or via clones.
- No more than one of the polygons is a grounded fill polygon.
- The distance over which the SMIN spacing is in violation is at least 10 times the WMIN value.

By default, the REPORT_SMIN_VIOLATION command is set to NO, in which case the *smin.sum* file is not created and the summary file does not contain warnings about SMIN violations.

It is important to have correct WMIN and SMIN values for any conductor whose definition contains an ETCH_VS_WIDTH_AND_SPACING table. The WMIN and SMIN values of the conductor should be at least as large as the smallest value (the first entry) in the WIDTHS and SPACINGS tables, respectively.

Inappropriate WMIN and SMIN values might cause opens or shorts of the neighboring layers by applying the etch values provided in the table, as follows:

- For entries in the WIDTHS table that are equal to the WMIN value, if positive etch values are greater than or equal to half of the WMIN value, the tool issues a warning message about possible opens.
- For entries in the SPACINGS table that are equal to the SMIN value, if the absolute value of the negative etch is greater than or equal to half of the SMIN value, a potential short condition exists. However, reporting of this condition is optional because most such errors should be caught during design rule checking. To enable SMIN violation reporting, set the REPORT_SMIN_VIOLATION command to YES.

Examples

For each SMIN violation, the *smin.sum* report lists the two nets involved, the bounding box, and the layer name. In the report, a net exhibits violations with respect to all of the nets whose names are indented in the subsequent lines. Each pair of nets might have multiple locations where an SMIN violation occurs, which is indicated in the report by multiple bounding boxes.

Every violation appears two times in the report, one time under each of the net names. The following lines are examples from an SMIN violation report:

```
SIGNALNET
  LEFT0
   at BBox=(911.11,-0.133),(880.455,-0.047) on M2,M2
   at BBox=(199.999,-0.133),(177.702,-0.047) on M2,M2
   ...
  RIGHT0
   at BBox=(977.234,0.047),(955.555,0.133) on M2,M2
   at BBox=(933.332,0.047),(910.185,0.133) on M2,M2
   ...

RIGHT0
  SIGNALNET
   at BBox=(977.234,0.047),(955.555,0.133) on M2,M2
   at BBox=(933.332,0.047),(910.185,0.133) on M2,M2
   ...
```

See Also

- ETCH_VS_WIDTH_AND_SPACING
- SMIN
- WMIN
- SMIN Violations Reports

Feedback

REPORT_UNMAPPED_GDS_OASIS_LAYERS

Specifies whether to provide warning messages for unmapped layers in a GDSII file or OASIS file.

Syntax

```
REPORT_UNMAPPED_GDS_OASIS_LAYERS: YES | NO
```

Arguments

Argument	Description
YES	Reports unmapped layers
NO (default)	Does not report unmapped layers

Description

If you set the REPORT_UNMAPPED_GDS_OASIS_LAYERS command to YES, the tool issues a warning message if either of the following conditions exist:

- A layer is present in a GDSII file specified with the GDS_FILE command but is not present in the mapping file specified with the GDS_LAYER_MAP_FILE command.
- A layer is present in an OASIS file specified with the OASIS_FILE command but is not present in the mapping file specified with the OASIS_LAYER_MAP_FILE command.

Examples

The following is an example of a warning message generated by the the REPORT_UNMAPPED_GDS_OASIS_LAYERS command:

```
WARNING: Missing layer mapping for layernum:datatype(1:0) in layer map
 file '../empty.gdsmap' (SX-2151)
```

See Also

- REPORT_UNMAPPED_GRD_LAYERS

Feedback

REPORT_UNMAPPED_GRD_LAYERS

Specifies whether to provide warning messages for unmapped layers in an ITF file.

Syntax

```
REPORT_UNMAPPED_GRD_LAYERS: YES | NO
```

Arguments

Argument	Description
YES	Reports unmapped layers
NO (default)	Does not report unmapped layers

Description

If you set the REPORT_UNMAPPED_GRD_LAYERS command to YES, the tool issues a warning message if either of the following conditions exist:

- A layer is present in an ITF file specified with the ITF_FILE command but is not present in the mapping file specified with the MAPPING_FILE command.
- A layer is present in an nxtgrd file specified with the TCAD_GRD_FILE command but is not present in the mapping file specified with the MAPPING_FILE command.

Examples

The following is an example of a warning message generated by the the REPORT_UNMAPPED_GRD_LAYERS command:

```
WARNING: Missing layer mapping for ITF layer 'POLY'. (SX-2156)
```

See Also

- REPORT_UNMAPPED_GDS_OASIS_LAYERS

Feedback

RES_UPDATE_FILE

Provides a method to modify resistance models without regenerating an nxtgrd file.

Syntax

```
RES_UPDATE_FILE: filename
```

Arguments

Argument	Description
filename	File that contains the resistance update data for metal layers and via layers

Description

Use the RES_UPDATE_FILE command to specify a file that contains RPSQ_VS_SI_WIDTH and RPSQ_VS_SI_WIDTH_AND_LENGTH tables for CONDUCTOR layers and RPV_VS_AREA values for VIA layers. These specifications take precedence over the mapping file and nxtgrd file settings for the corresponding ITF layers.

Each layer can be defined only one time in the update file. Multiple layer resistance definitions are treated as errors.

The layers defined in the file specified by the RES_UPDATE_FILE command are not database layers. They are ITF layers that can apply to several database layers in the layout, as specified by the mapping file.

Note:

The RES_UPDATE_FILE command cannot be used with simultaneous multicorner extraction. Remove the SIMULTANEOUS_MULTI_CORNER command from the command file if you plan to use the RES_UPDATE_FILE command.

Examples

The following command specifies a file named res.itf that defines resistance update data for metal layers M1 and GATE and via layer via1.

```
RES_UPDATE_FILE: res.itf
```

The res.itf file contains the RPSQ_VS_SI_WIDTH and RPV_VS_AREA tables for the layers that need to be modified, as follows:

```
CONDUCTOR M1 {
RPSQ_VS_SI_WIDTH {
  (0.34, 0.075) (0.40, 0.062)
  (0.823, 0.0817)(2.0, 0.0321)
```

```
   (6.0, 0.0173)
   }
}
CONDUCTOR GATE {
RPSQ_VS_SI_WIDTH_AND_LENGTH {
  LENGTHS {0.05 0.1 0.15}
  WIDTHS {0.02 0.022 0.024}
  VALUES {0.1 0.23 0.45
          0.12 0.26 0.47
          0.18 0.28 0.53 }
  }
}
VIA via1 {
RPV_VS_AREA {
  (350, .5)
  (600, .25)
  (200, .5)
  }
}
```

See Also

- RPSQ_VS_SI_WIDTH
- RPSQ_VS_SI_WIDTH_AND_LENGTH
- RPV_VS_AREA

Feedback

RETAIN_CAPACITANCE_CAP_MODELS

Specifies model-specific plate-to-plate capacitance retention for capacitor devices.

Syntax

```
RETAIN_CAPACITANCE_CAP_MODELS: model_list
```

Arguments

Argument	Description
model_list	List of capacitor devices for which plate-to-plate capacitance is to be retained. Accepted model names in the list include either the schematic or layout names, regardless of the XREF command setting. Default: none

Description

In certain applications, it is advantageous to retain parasitic capacitances within the parasitic netlist for capacitor devices, particularly when you want to simulate devices using parasitic capacitances instead of device simulation models. To do this, you can selectively retain plate-to-plate capacitance for designed capacitor devices. Devices whose capacitances are to be retained are specified by model name.

Specify a list of model names of capacitor devices for which IGNORE_CAPACITANCE statements should not be generated between terminal layers and for which plate-to-plate capacitance should not be ignored by the StarRC extraction engine.

If a specified model name matches the schematic model name of a capacitor device in the design, the RETAIN_CAPACITANCE_CAP_MODELS functionality is propagated to all layout capacitor devices matching that schematic model name. If a specified model name matches the layout model name of a capacitor device in the design, the RETAIN_CAPACITANCE_CAP_MODELS functionality is propagated only to layout capacitor devices matching that layout model name. All of this occurs independent of the XREF command in the command file.

Note:

If the MODEL_TYPE command is not specified in the command file, the default setting is LAYOUT.

Errors

A warning is issued when a model name exists in the RETAIN_CAPACITANCE_CAP_MODELS command for which no corresponding capacitor exists.

Feedback

Examples

The following command retains capacitance for device cm1m2:

```
RETAIN_CAPACITANCE_CAP_MODELS: cm1m2
```

See Also

- MODEL_TYPE

Feedback

RETAIN_FLOATING_NETS

Specifies floating nets to be exempted from floating net handling commands.

Syntax

```
RETAIN_FLOATING_NETS: netnames
```

Arguments

Argument	Description
netnames	List of floating net names, delimited by white space Default: none

Description

The RETAIN_FLOATING_NETS command specifies a list of floating nets that should be reported in the output netlist (or GPD) regardless of the settings of the REMOVE_FLOATING_NETS and TRANSLATE_FLOATING_AS_FILL commands.

The *, ?, and ! wildcards are accepted. Case sensitivity is determined by the CASE_SENSITIVE command. The net name must match the type specified by the NET_TYPE command, either LAYOUT or SCHEMATIC.

The RETAIN_FLOATING_NETS command takes precedence over the NETS command. Consider the following example:

```
NETS: * !ab*
REMOVE_FLOATING_NETS: YES
RETAIN_FLOATING_NETS: ab25 ab_m2
```

In this case, floating nets ab25 and ab_m2 are included in the output; all other floating nets and all other nets whose names begin with “ab” are not included.

See Also

- REMOVE_FLOATING_NETS
- TRANSLATE_FLOATING_AS_FILL

Feedback

RETAIN_GATE_CONTACT_COUPLING

Retains gate-to-contact coupling capacitance in transistor-level flows.

Syntax

```
RETAIN_GATE_CONTACT_COUPLING: YES | NO
```

Arguments

Argument	Description
YES	Retains (does not ground) the gate-to-contact capacitance
NO (default)	Does not retain the gate-to-contact capacitance

Description

By default, the StarRC tool grounds all coupling capacitances. However, several StarRC commands allow you to ground or retain specific types of coupling capacitances. The RETAIN_GATE_CONTACT_COUPLING command affects the gate-to-contact capacitance in the Hercules, IC Validator, and Calibre flows.

You can retain the gate-to-contact capacitance by using either of the following two methods:

1. Set the COUPLE_TO_GROUND command to YES RETAIN_GATE_CONTACT_COUPLING (or to YES RETAIN_GATE_CONTACT_AND_DIFFUSION_COUPLING) and the EXTRACT_VIA_CAPS command to YES. In this case, the gate-to-contact capacitance is retained, regardless of the settings of the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands.
2. Set the RETAIN_GATE_CONTACT_COUPLING command to YES and the EXTRACT_VIA_CAPS command to YES. In this case,
   - If the COUPLE_TO_GROUND command is set to NO, the gate-to-contact capacitance is retained only if the capacitance meets the thresholds specified by the COUPLING_ABS_THRESHOLD and COUPLING_REL_THRESHOLD commands.
   - If the COUPLE_TO_GROUND command is set to YES, the gate-to-contact capacitance is always retained.

The RETAIN_GATE_CONTACT_COUPLING command takes precedence. If you set the RETAIN_GATE_CONTACT_COUPLING command to NO and the COUPLE_TO_GROUND command to YES RETAIN_GATE_CONTACT_COUPLING, the COUPLE_TO_GROUND setting is reset to YES and the gate-to-contact capacitance is not retained.

Feedback

By default, the StarRC tool does not retain coupling capacitances for power nets. To retain the gate-to-contact capacitance for power nets, use one of these command settings:

- Set the `POWER_EXTRACT` command to `YES` to retain the gate-to-contact capacitances for all signal and power nets.
- Set the `POWER_EXTRACT` command to `DEVICE_LAYERS` to retain the gate-to-contact capacitances for all signal nets and all power nets whose layer are specified with the `device_layers` keyword in a `conducting_layers` statement in the mapping file.

Note:

If the device SPICE models already contain gate-to-contact capacitances, you would typically avoid capacitance double counting by setting the `EXTRACT_VIA_CAPS` command to `YES IGNORE_GATE_CONTACT_COUPLING`. In this case, the tool does not extract the gate-to-contact capacitances; therefore, the commands discussed in this section have no effect on these capacitances.

See Also

- COUPLE_TO_GROUND
- COUPLING_MULTIPLIER
- COUPLING_ABS_THRESHOLD
- COUPLING_REL_THRESHOLD
- EXTRACT_VIA_CAPS

Feedback

RING_AROUND_THE_BLOCK

Generates a virtual ring on every routing layer around the block.

Syntax

```
RING_AROUND_THE_BLOCK: YES | NO
```

Arguments

Argument	Description
YES	Generates a ring around the block
NO (default)	Does not generate a ring around the block

Description

The RING_AROUND_THE_BLOCK command generates a virtual ring around the block to be extracted, which is specified with the BLOCK command. The capacitance between the block material and the imaginary ring is calculated as though the block is surrounded by solid metal.

If you use the RING_AROUND_THE_BLOCK command for a LEF/DEF flow, the BLOCK_BOUNDARY command is also required.

If you use the RING_AROUND_THE_BLOCK command for a Milkyway flow, it is not necessary to use the BLOCK_BOUNDARY command because the boundary information can be read directly. However, you can use the BLOCK_BOUNDARY command to override the design database boundary information.

Block material that lies outside the specified boundary does not create shorts with or attach capacitance from the imaginary rings, even if they overlap. Overlaps should be avoided, because shielding of nets occurs.

You can specify the spacing between the block boundary and the ring with the RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER command.

The rings are grounded by default but can be given a special name with the ZONE_COUPLE_TO_NET command.

See Also

- BLOCK
- BLOCK_BOUNDARY

Feedback

- RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER
- ZONE_COUPLE_TO_NET

Feedback

RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER

Specifies the spacing between the block boundary and the ring around the block in hierarchical analysis.

Syntax

```
RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER: multiplier
```

Arguments

Argument	Description
multiplier	Multiplier; floating-point number greater than or equal to 0.5 Units: none Default: 0.5

Description

The RING_AROUND_THE_BLOCK_SMIN_MULTIPLIER command specifies the spacing between the block boundary and the ring around the block. The spacing is equal to the product of the multiplier and the SMIN value of the nxtgrd layer. The SMIN value might be different from layer to layer, resulting in different ring spacing.

See Also

- BLOCK
- BLOCK_BOUNDARY
- RING_AROUND_THE_BLOCK

Feedback

SELECTED_CORNERS

Specifies the corners to be extracted in the simultaneous multicorner flow.

Syntax

```
SELECTED_CORNERS: name_list
```

Arguments

Argument	Description
name_list	The corners to be extracted, separated by spaces

Description

This command specifies the corners to be extracted in the simultaneous multicorner (SMC) flow. The command has an effect only if the SMC flow is enabled.

Every corner listed in the SELECTED_CORNERS command must be defined in the corners file specified by the CORNERS_FILE command.

To create a SPEF netlist containing more than one corner, use a colon (:) between corner names. The first corner listed is considered the primary corner for output netlist reduction. This feature is valid only for transistor-level flows.

The name of the netlist file for a given corner is the base file name specified in the NETLIST_FILE command, with the corner name appended to it.

Examples

Consider the following definition of selected corners:

```
SELECTED_CORNERS: C1 C3 C4
```

In this case, three netlists are created. The first netlist contains parasitic data from C1, the second netlist contains data from C3, and the third netlist contains data from C4.

See Also

- CORNERS_FILE
- SIMULTANEOUS_MULTI_CORNER
- Simultaneous Multicorner Extraction
- DEFAULT_CORNER

Feedback

SHEET_COUPLE_TO_NET

Specifies a prefix net name for the sheet zone extraction capability. This user-defined name appears in the generated output netlist.

Syntax

```
SHEET_COUPLE_TO_NET: prefix_name
```

Arguments

Argument	Description
prefix_name	Net prefix name reported in the output netlist Default: none

Description

If this command is not specified, the prefix name is set to `ZONE_SHEET`.

Examples

```
METAL_SHEET_OVER_AREA: METAL2 0 0 100 100
METAL_SHEET_OVER_AREA: METAL2 200 200 400 400
METAL_SHEET_OVER_AREA: METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL: YES
```

See Also

- METAL_SHEET_OVER_AREA
- SHEET_COUPLE_TO_NET_LEVEL

Feedback

SHEET_COUPLE_TO_NET_LEVEL

Enables or disables a net name suffix using the layer-level number for the sheet zone netlist capability.

Syntax

```
SHEET_COUPLE_TO_NET_LEVEL: NO | YES
```

Arguments

Argument	Description
NO (default)	Disables net name suffix reporting in sheet zone netlist output
YES	Enables net name suffix reporting in sheet zone netlist output

Description

Enables or disables a net name suffix using the layer-level number for the sheet zone netlist capability.

Examples

```
METAL_SHEET_OVER_AREA: METAL2 0 0 100 100
METAL_SHEET_OVER_AREA: METAL2 200 200 400 400
METAL_SHEET_OVER_AREA: METAL4 0 0 100 100
SHEET_COUPLE_TO_NET: zone_sheet
SHEET_COUPLE_TO_NET_LEVEL: YES
```

See Also

- METAL_SHEET_OVER_AREA
- SHEET_COUPLE_TO_NET

Feedback

SHORT_EQUIV_NODES

Specifies whether to add a shorting resistor between short equivalent nodes.

Syntax

```
SHORT_EQUIV_NODES: YES | NO
```

Arguments

Argument	Description
YES	Shorts equivalent nodes
NO (default)	Does not short equivalent nodes

Description

The SHORT_EQUIV_NODES command merges equivalent nets into one single net. A shorting resistor is added during netlist creation to merge the nets.

You must specify the XREF: YES command to use the SHORT_EQUIV_NODES command.

Note:

The resistance value is set to 0.01 ohm for shorting opens and 0.001 ohm for shared nodes.

You can use this command when using the IC Validator or Hercules LVS tools. You cannot use this command with the Calibre Connectivity Interface because this database does not contain information about equivalent nodes.

Examples

When you use the following command in the StarRC command file, the tool adds a shorting resistor between short equivalent nodes:

```
SHORT_EQUIV_NODES : YES
```

See Also

- XREF

Feedback

SHORT_PINS

Specifies whether to short top-level pin ports that have multiple placements.

Syntax

```
SHORT_PINS: YES | NO | MIXED
```

Arguments

Argument	Description
YES (default)	Reports all of the placements of the pin port as a single node (a single *|P)
NO	Each marker group (IC Validator or Hercules flows) or PIN at the top level (NDM, Milkyway, or LEF/DEF flows) is uniquely identified with a suffix defined by the NETLIST_RENAME_PORTS command.
MIXED	Same-name pins are reported as a single node (a single *|P); if pin names are different, they are listed as separate *|P entries. In NDM, Milkyway, or LEF/DEF flows, ensures correct back-annotation of a parasitic netlist.

Description

Accurate back-annotation of a parasitic netlist to a design requires that the naming conventions be understood. In some designs, physical nets can contain multiple physical pins, whose names might be the same or different.

When you use the PIN_CUT_THRESHOLD command, the StarRC tool internally sets the short_pins command to NO.

A unique pin instance is defined as a physically connected group of objects marked as interface material: a PIN section in a DEF file, a PIN type in a Milkyway or NDM design, or a MARKER_LAYER instance in a Hercules flow. Each group of connected objects is a pin instance.

By default, or if you specify the SHORT_PINS:YES command, the *P object connected to a net always inherits the name of the *D_NET object when the pins have multiple names.

If the layout database has unique names for each pin instance associated with a single port, that information is used to write the *P object in the netlist, instead of using the NETLIST_RENAME_PORTS command. The delimiter specified in the NETLIST_RENAME_PORTS command is used only if no naming information exists in the input layout database. The default is a single underscore character followed by sequential numbering.

Feedback

The effect of the SHORT_PINS command on the *P names is shown in Table 87 for different combinations of net and pin names.

Table 87 Effect of the SHORT_PINS Command

Net name	Logical pins	Physical pins	SHORT_PINS: YES Name of *P	SHORT_PINS: NO Name of *P	SHORT_PINS: MIXED Name of *P
CLK	CLK	CLK	CLK	CLK	CLK
CLK	CLK	CLK (multiple)	CLK	CLK CLK_1 ...	CLK
CLK	PX	PX	CLK	PX	PX
CLK	PX CLK	PX CLK	CLK	PX CLK	PX CLK
CLK	PX CLK	PX CLK (multiple)	CLK	PX CLK CLK_1 ...	PX CLK
CLK	PX CLK	PX (multiple) CLK (multiple)	CLK	PX PX_1 ... CLK CLK_1 ...	PX CLK

Fixing Redundant Ports

Redundant ports for an HSIM or NanoSim parasitic back-annotation flow can be removed. The port name postfix "_1" in an extracted file is generated because the input database contained partial unique naming information, such as:

```
SUBCKT v_ctl VCI_P0 VCI_P0_1 GND_P0 GND_P0_1 VDD_P0 VDD_P0_1
```

The redundant ports "VCI_P0_1", "GND_P0_1" and "VDD_P0_1" can be removed by changing the StarRC command SHORT_PINS:NO to SHORT_PINS:YES.

The result is as follows:

```
SUBCKT v_ctl VCI_P0 GND_P0 VDD_P0
```

The extracted standard parasitic format file can then be used for HSIM or NanoSim parasitic back-annotation.

Examples

Example 1. Design with unique pin names

Contents of the DEF file:

```
PINS 2 ;
   - C.1 + NET C  + LAYER METAL2 ( 0 0 ) ( 1000 1600 )
      + PLACED  ( 263800 136000 ) N ;
```

```
   - C.2 + NET C  + LAYER METAL2 ( 0 0 ) ( 1000 1600 )
      + PLACED  ( 264800 136000 ) N ;
   END PINS
```

Contents of the parasitic netlist for different settings of the SHORT_PINS command:

```
   SHORT_PINS: NO
   *|NET C 0.0295887PF
   *|P (C.2 B 0 264.8 136)
   *|P (C.1 B 0 263.8 136)

   SHORT_PINS: YES
   *|NET C 0.0295887PF
   *|P (C B 0 264.8 136)
```

Example 2. Design with multiple pins that have the same name

Contents of the DEF file:

```
PINS 2 ;
    - C + NET C + LAYER METAL2 ( 0 0 ) ( 1000 1600 )
       + PLACED  ( 263800 136000 ) N ;
   - C + NET C + LAYER METAL2 ( 0 0 ) ( 1000 1600 )
      + PLACED  ( 264800 136000 ) N ;
   END PINS
```

Contents of the parasitic netlist for different settings of the SHORT_PINS command:

```
   SHORT_PINS: NO, NETLIST_RENAME_PORTS: _
   *|NET C 0.0295887PF
   *|P (C_2 B 0 264.8 136)
   *|P (C_1 B 0 263.8 136)

   SHORT_PINS: YES
   *|NET C 0.0295887PF
   *|P (C B 0 264.8 136)
```

Example 3. Design with mixed naming styles

Contents of the DEF file:

```
PINS 3 ;
   - C.1 + NET C  + LAYER METAL2 ( 0 0 ) ( 1000 700 )
      + PLACED  ( 263800 136000 ) N ;
   - C.1 + NET C  + LAYER METAL2 ( 0 0 ) ( 1000 700 )
      + PLACED  ( 263800 137000 ) N ;
   - C.2 + NET C  + LAYER METAL2 ( 0 0 ) ( 1000 1600 )
      + PLACED  ( 264800 136000 ) N ;
   END PINS
```

Contents of the parasitic netlist for different settings of the SHORT_PINS command:

```
SHORT_PINS: NO, NETLIST_RENAME_PORTS: _
*|NET C 0.0295974PF
*|P (C.1 B 0 263.8 136)
*|P (C.2 B 0 264.8 136)
*|P (C.1_1 B 0 263.8 137)

SHORT_PINS: YES
*|NET C 0.0295887PF
*|P (C B 0 264.8 136)
```

See Also

- NETLIST_RENAME_PORTS

Feedback

SHORT_PINS_IN_CELLS

Merges all electrically equivalent pins into a single port for the specified list of skip cells.

Syntax

```
SHORT_PINS_IN_CELLS: list
```

Arguments

Argument	Description
list	The list of skip cell names Default: *

Description

Place-and-route databases sometimes contain ports that are electrically equivalent. These ports are logically equivalent but can have a wide geographic distribution throughout the design. The StarRC tool merges all electrically equivalent pins into a single port for every skip cell on the SHORT_PINS_IN_CELLS list.

Occasionally, designs with electrically equivalent ports are instantiated as macros. If a macro with electrically equivalent ports is on the skip cells list, by default, the tool reports a single connection for the electrically equivalent port in the netlist, using the first electrically equivalent pin location. Because the electrically equivalent pins can be so widely distributed in the layout, it can be desirable to treat each of the electrically equivalent pins as a unique port for simulation. Negating a cell from the SHORT_PINS_IN_CELLS list means that the StarRC tool treats each electrically equivalent pin as a unique port and uses the original database port suffix to report it.

Wildcards “*”, “?”, and “!” are acceptable. This command can be specified multiple times.

See Also

- CASE_SENSITIVE
- SHORT_PINS
- SKIP_CELLS

Feedback

SHORTS_LIMIT

Limits the number of shorts to include in summary reports.

Syntax

```
SHORTS_LIMIT: max_count
```

Arguments

Argument	Description
max_count	Maximum number of shorts to report Default: 1000

Description

In cases where the StarRC tool identifies a large number of shorts, you might want to limit the amount of detail included in summary reports and the number of times that similar messages are issued.

The `SHORTS_LIMIT` command specifies the maximum number of unique shorts for which to report detailed information such as layer names and bounding boxes in the shorts_all.sum file. If the limit is exceeded, the StarRC tool writes a warning message in the file and does not report the additional shorts.

The shorts controlled by this command do not include fill shorts. For fill shorts, use the `FILL_SHORTS_LIMIT` command.

See Also

- FILL_SHORTS_LIMIT
- Shorts Reports

Feedback

SIMULTANEOUS_MULTI_CORNER

Enables the simultaneous multicorner (SMC) flow.

Syntax

```
SIMULTANEOUS_MULTI_CORNER: YES | NO
```

Arguments

Argument	Description
YES (default)	Enables the simultaneous multicorner flow
NO	Uses the single-corner extraction flow

Description

Simultaneous multicorner (SMC) extraction optimizes the efficient extraction of multiple process and temperature corners for a single design. The SMC flow is enabled by default for both gate-level and transistor-level extraction.

To use simultaneous multicorner extraction, the CORNERS_FILE and SELECTED_CORNERS commands must also be set. Table 88 summarizes the effect of StarRC command settings on the extraction mode.

Table 88 Effect of Command File Settings on SMC Extraction

SIMULTANEOUS_MULTI_CORNER command	CORNERS_FILE and SELECTED_CORNERS commands	Extraction
YES	present	SMC
not present	present	SMC
NO	not present	single-corner
not present	not present	single-corner
YES	not present (either or both)	error
NO	present (either or both)	error

For each corner, the corners file must define the corner name, the process corner (through the nxtgrd file), and the operating temperature. During SMC extraction,

- The StarRC tool uses the nxtgrd files specified in the corners file and ignores any other TCAD_GRD_FILE commands in the command file
- The tool uses the operating temperatures specified in the corners file and ignores any other OPERATING_TEMPERATURE commands in the command file

Using the Field Solver Flows With Simultaneous Multicorner Extraction

If you use the SMC flow with the EXTRACTION: FSCOMPARE command, the StarRC tool generates the total and coupling capacitance report files for each corner with a unique nxtgrd file specified by the SELECTED_CORNERS command. If multiple corners with the same nxtgrd file are selected, the tool generates a report only for the first corner because capacitance does not depend on the operating temperature.

You can use the FS_EXTRACT_NETS command in the simultaneous multicorner flow. If you specify nets with the SELECTED_CORNERS and FS_EXTRACT_NETS commands, the field solver generates netlist files for the different corners.

Examples

The following example uses one netlist per corner in a simultaneous multicorner flow. The three corners are the nxtgrd file named nominal.nxtgrd analyzed at two temperatures (-25 and 125°C) and the nxtgrd file named rcmax.nxtgrd analyzed at one temperature (25°C).

The command file contains the following commands:

```
SIMULTANEOUS_MULTI_CORNER: YES
CORNERS_FILE: corners.smc
SELECTED_CORNERS: NOM_T1 NOM_T2 RCMAX_T3
```

The corners file contains the following commands:

```
CORNER_NAME: NOM_T1
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: -25

CORNER_NAME: NOM_T2
TCAD_GRD_FILE: nominal.nxtgrd
OPERATING_TEMPERATURE: 125

CORNER_NAME: RCMAX_T3
TCAD_GRD_FILE: rcmax.nxtgrd
OPERATING_TEMPERATURE: 25
```

The resulting output files are star_NOM_T1.spf, star_NOM_T2.spf, and star_RCMAX_T3.spf.

Feedback

See Also

- CORNERS_FILE
- SELECTED_CORNERS
- Simultaneous Multicorner Extraction

Feedback

SKIP_CELL_AGF_FILE

Imports Hercules annotated GDSII (AGF) files as the detail view for skip cells.

Syntax

```
SKIP_CELL_AGF_FILE: cell agf_file herc_ideal_netlist
```

Arguments

Argument	Description
cell	AGF cell name
agf_file	AGF file name
herc_ideal_netlist	Hercules ideal netlist name

Description

Use this command in the StarRC command file to import Hercules annotated GDSII (AGF) files as the detail view for skip cells. Specify the command one time for each skip cell that has an AGF description.

- A SKIP_CELL_AGF_FILE command cannot be specified as a macro.
- All SKIP_CELL_AGF_FILE commands specified must use the same layer mapping as defined in the GDS_LAYER_MAP_FILE command.
- A layer mapping file must also be shared with the GDS_FILE (if it is used).

If the list of cells in a COUPLE_NONCRITICAL_NETS command contains a SKIP_CELL_AGF_FILE cell (or descendant), the layout names are used to identify its contents. If the list of cells in a COUPLE_NONCRITICAL_NETS command does not contain a SKIP_CELL_AGF_FILE cell, its contents are annotated as noncritical (ground potential).

Child cells of a COUPLE_NONCRITICAL_NETS command must be specified in the COUPLE_NONCRITICAL_NETS command as well; they are not automatically included.

Examples

```
SKIP_CELL_AGF_FILE: INV_BUF buf.agf buf.sp
```

See Also

- COUPLE_NONCRITICAL_NETS
- GDS_LAYER_MAP_FILE

Feedback

SKIP_CELL_PORT_PROP_FILE

Specifies files containing pin or port information for skip cell properties.

Syntax

```
SKIP_CELL_PORT_PROP_FILE: file1 [file2 … fileN]
```

Arguments

Argument	Description
fileN	File name Default: none

Description

Use the SKIP_CELL_PORT_PROP_FILE command to specify files that contain pin information (such as direction or capacitance) for skip cells in LEF/DEF, Milkyway, and NDM format Fusion Compiler or IC Compiler II design databases. Valid for transistor-level IC Validator and Calibre Connectivity Interface flows, where

- Wildcards are supported.
- With the XREF:YES command, the INSTANCE_PORT: PORT_DIR command is supported to apply extraction mode only to ports with the specified direction.

The description of all port properties of a cell should be in one CELL block and should be uniquely defined. For example, do not specify multiple descriptions of a cell in the same library or another reference library. Ensure that reference libraries you specify contain unique definitions of cells.

The port property file format is as follows. All parameters are required.

```
CELL cell1name
pin1name pin1io pin_cap
pin2name pin2io pin_cap

CELL cell2name
pin1name pin1io pin_cap
pin2name pin2io pin_cap
```

Table 89 describes the file parameters.

Table 89 Port Property File Parameters

Argument	Description
CELL	Keyword specifying the beginning of the cell definition

Feedback

Table 89 Port Property File Parameters (Continued)

Argument	Description
cell_name	The cell name
pin_name	The pin or port name
pin_io	Pin or port direction. Valid values are I, O, or B representing input, output, and bidirectional; also supported in Calibre Connectivity Interface flow.
pin_cap	The pin capacitance value, which can include a unit. Valid units are: FF, PF, NF, uF and mF.

For a net attached to a pin, the pin capacitance does not affect the net capacitance in the *|NET or D_NET sections of a netlist. Therefore there is no risk of pin capacitance double counting by downstream tools.

However, the pin capacitance might affect smart capacitance decoupling because it is included in the total net capacitance for the threshold calculation specified by the `COUPLING_REL_THRESHOLD` command.

IC Validator and Calibre Connectivity Interface Flows

You should specify the port directions in a port property file to use port information in the IC Validator and Calibre Connectivity Interface (CCI) flows. This is needed as the port information is not available in the input database. The port direction specified with the `SKIP_CELL_PORT_PROP_FILE` command as follows is used by the `INSTANCE_PORT` and `INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER` commands.

```
SKIP_CELL_PORT_PROP_FILE: cellprop.txt
```

The cellprop.txt file should include information of all the output ports for all the cells, as shown in the following example. Note that you can specify only one port in a line and wildcards are allowed.

```
CELL abc*
mfcp* O
abc* O
CELL cc0*
o* O
```

Table 90 lists the behavior of the `SKIP_CELL_PORT_PROP_FILE` command when you use a wildcard in a cell model. Where, the wildcard in a cell model applies the following rule: the explicit non wildcard cell model name is changed to a wildcard name with the * symbol as the default. For example, the non wildcard xya6000abc string is changed to xya6000*.

Feedback

Table 90 Using wildcard in a cell model

SKIP_CELL_PORT_PROP_FILE	Behavior
`CELL *` `abcport1 O`	`CELL *` is not allowed and the StarRC tool issues an error message.

`CELL xya6000*`
`abcport* O`
	The command sets other ports (`<other ports> I`) as bidirectional (`B`) that can be overidden with the `SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR` command.
	`abcport1 O`
	`abcport2 O`
	`<other ports> I`

Examples

The following is an example of a port property file:

```
CELL XYZ
pin1 I 20FF
pin2 O 35FF
```

See Also

- COUPLING_REL_THRESHOLD
- INSTANCE_PORT
- INSTANCE_PORT_REPORT_FILE
- INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER
- SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR

Feedback

SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR

Sets a default port direction for all skip cell ports.

Syntax

```
SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR: B | I | O
```

Arguments

Argument	Description
`B` (the default)	Sets the port direction of the skip cell ports to bidirectional.
`I`	Sets the port direction of the skip cell ports to input.
`O`	Sets the port direction of the skip cell ports to output.

Description

Use the `SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR` command to set a default port direction for all skip cell ports, except for the port names defined with the `SKIP_CELL_PORT_PROP_FILE` command.

Examples

The following command shows how to set all the skip cell output ports to bidirectional, except for the ports defined in the cellprop.txt port property file.

```
# sets all instance ports to CONDUCTIVE
INSTANCE_PORT: CONDUCTIVE

# sets all input ports to be SUPERCONDUCTIVE for IC Validator and CCI
# gate-level flows
INSTANCE_PORT: SUPERCONDUCTIVE PORT_DIR IN

# specifies port directions in the cellprop.txt port property file
SKIP_CELL_PORT_PROP_FILE: cellprop.txt

# for example, the port property file can have the following
# user-defined port direction settings:
CELL XYZ
o* O
z* O

# sets all the skip cell output ports to bidirectional,
# except fot the ports defined in the cellprop.txt file
SKIP_CELL_PORT_PROP_FILE_DEFAULT_PORT_DIR: B
```

Feedback

See Also

- INSTANCE_PORT
- INSTANCE_PORT_REPORT_FILE
- INSTANCE_PORT_LOCATION_CLOSER_TO_DRIVER
- SKIP_CELL_PORT_PROP_FILE

Feedback

SKIP_CELL_SOURCE_REPORT_FILE

Specifies a file that lists the source file used for skip cells in the design. Valid only for gate-level flows.

Syntax

```
SKIP_CELL_SOURCE_REPORT_FILE: source_report
```

Arguments

Argument	Description
source_report	A file that contains source information about skip cells Default: none

Description

The SKIP_CELL_SOURCE_REPORT_FILE command specifies the name of a file in which the StarRC tool should list the source file used for all skip cells in the design. This command supports LEF/DEF and NDM format designs.

If this command is not used, the tool writes the information in the summary file. If the command is used, the tool writes the information in the specified file and does not write the information in the summary file. In this report file, the StarRC tool ignores any limits set by the MESSAGE_SUPPRESSION or MESSAGE_SUPPRESSION_LIMIT commands.

The file contains lines in the following format:

```
messageID celltype 'name1' applied to LEF/NDM cell 'name2' from 'file'
```

For example:

```
SX-2170 GDSCell 'abc' applied to LEF/NDM cell 'xyz' from 'block23.lef'
SX-2171 OASISCell 'a02' applied to LEF/NDM cell 'b45' from 'design_A7'
```

In addition, the tool writes warning messages about missing or duplicated cells into the report file.

See Also

- SKIP_CELLS
- MESSAGE_SUPPRESSION
- MESSAGE_SUPPRESSION_LIMIT

Feedback

SKIP_CELLS

Creates a white-space-delimited list of cells to skip during extraction.

Syntax

```
SKIP_CELLS: cell1 cell2 … cellN
```

Arguments

Argument	Description
cell1 cell2 … cellN	A list of cells to be skipped during extraction. All instances of these cells are skipped. Default: *

Description

The SKIP_CELLS command creates a white-space-delimited list of cells for to skip during extraction. This command can be specified multiple times in a single command file. The asterisk (*), exclamation mark (!), and question mark (?) wildcard characters are acceptable. Case sensitivity for selection purposes is determined by the value of the CASE_SENSITIVE command, but the netlist always retains the case of the original input database.

Skip cells are typically cells with their own timing models, which can later be applied, along with the StarRC parasitic netlist, to perform a timing analysis. Skip cells should contain labeled or otherwise annotated pin shapes.

The default for all extraction flows is to skip all lower-level blocks in the input database, so any macro blocks without timing models must be negated from the list.

To skip only specific instances of a cell, use the SKIP_INSTANCES command.

The translate.sum file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a SKIP_CELLS or SKIP_PCELLS command. Skipped instances are not reported in this file.

Note:

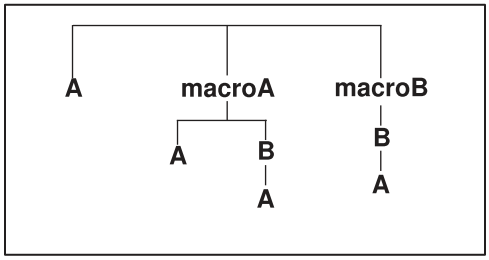
When you use the CALIBRE_PDBA_FILE command, the SKIP_CELLS command might fail because the DFM properties annotation can promote the instance hierarchy.

Examples

```
SKIP_CELLS: cellA !cellB cell? *C …
SKIP_CELLS: * !macroA !macroB …
```

Feedback

Figure 218 Example Using SKIP_CELLS



Using the hierarchy shown in Figure 218, only cell A is skipped in the following example:

```
SKIP_CELLS: A
```

Cell macroA is not skipped. The resulting netlist contains 4 instances of A.

If you use the following command, cells A and B is skipped:

```
SKIP_CELLS: A B
```

Cells macroA and macroB are not skipped. The resulting netlist contains 2 instances of A and 2 instances of B.

In following example, all cells are skipped:

```
SKIP_CELLS: *
```

The following example specifies that all cells except macroA and macroB are skipped:

```
SKIP_CELLS: * !macro?
```

In following example, no cells are skipped:

```
SKIP_CELLS: !*
```

See Also

- CASE_SENSITIVE
- CELL_TYPE
- SKIP_INSTANCES
- INSTANCE_TYPE

Feedback

SKIP_CELLS_COUPLE_TO_NET

Specifies a lump net for all coupling capacitance between top-level routes and noncritical skip cell material.

Syntax

```
SKIP_CELLS_COUPLE_TO_NET: net_name
```

Arguments

Argument	Description
net_name	The net name to the noncritical skip cell material Default: 0 (ground)

Description

The default is to use ground as the noncritical skip cell material potential.

You must set the `NETLIST_FORMAT: SPEF` and `COUPLE_TO_GROUND: NO` commands to use this option.

Use this option in conjunction with the `SKIP_CELLS_COUPLE_TO_NET_LEVEL` command to append the net name specified in the `SKIP_CELLS_COUPLE_TO_NET` command to the actual skip cell object layer number.

Examples

```
SKIP_CELLS_COUPLE_TO_NET: LUMP
```

See Also

- COUPLE_TO_GROUND
- NETLIST_FORMAT
- SKIP_CELLS_COUPLE_TO_NET_LEVEL

Feedback

SKIP_CELLS_COUPLE_TO_NET_LEVEL

Appends the SKIP_CELLS_COUPLE_TO_NET name to the real layer number for the SKIP_CELLS material.

Syntax

```
SKIP_CELLS_COUPLE_TO_NET_LEVEL: YES | NO
```

Arguments

Argument	Description
YES	Specifies that the layer number of the SKIP_CELLS material is appended to the assigned net name
NO (default)	Specifies that the layer number of the SKIP_CELLS material is not appended to the assigned net name

Description

This command appends the SKIP_CELLS_COUPLE_TO_NET name to the real layer number for the SKIP_CELLS material.

This command is ignored unless the SKIP_CELLS_COUPLE_TO_NET command is specified.

Examples

If the net name is LUMP and this command is set to YES, the resulting coupling capacitor between top-level net1 and a metal1 object inside a SKIP_CELLS might be as follows:

```
*CAP
…
12 net1 LUMP_1 1.2e-15
…
*END
```

See Also

- SKIP_CELLS_COUPLE_TO_NET

Feedback

SKIP_CELLS_FILE

Specifies a file containing SKIP_CELLS commands.

Syntax

```
SKIP_CELLS_FILE: file1 file2 …
```

Arguments

Argument	Description
file1, file2 ...	Files containing the defined skip cells Default: none

Description

This command specifies a file containing SKIP_CELLS commands. See the SKIP_CELLS command for more details.

The translate.sum file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a SKIP_CELLS or SKIP_PCELLS command.

Examples

The following lines appear in a file named cells_to_skip.dat.

```
SKIP_CELLS: cellA cellB cellX
SKIP_CELLS: cellC
```

The following command appears in the StarRC command file:

```
SKIP_CELLS_FILE: cells_to_skip.dat
```

As a result, cellA, cellB, cellC, and cellX are all skipped during extraction, and the cell names are written into the summary/translate.sum file.

See Also

- CASE_SENSITIVE
- SKIP_CELLS

Feedback

SKIP_INSTANCES

Specifies layout instances to skip.

Syntax

```
SKIP_INSTANCES instance_list
```

Arguments

Argument	Description
instance_list	The list of instance names to skip Default: none

Description

The SKIP_INSTANCES command lists layout instance names that should be treated as skip cells. Wildcards “*”, “!”, and “?” are accepted, but cannot be used as global arguments (for example, the SKIP_INSTANCES:* command is not allowed).

Instances that you might want to skip are those that already have timing model or parasitic netlists elsewhere in the library that represents the cell master.

The default is not to skip any specific instances. The SKIP_CELLS command has higher priority than the SKIP_INSTANCES command. In other words, all instances of cells listed in the SKIP_CELLS command are skipped, regardless of the setting of the SKIP_INSTANCES command.

The INSTANCE_TYPE command specifies whether the names in the SKIP_INSTANCE command are schematic names or layout names.

The translate.sum file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a SKIP_CELLS or SKIP_PCELLS command. Skipped instances are not reported in this file.

Examples

In this example, all macroA instances are flattened, except macroA_instance1. All macroB instances are flattened, except macroB_instance1.

```
SKIP_CELLS: * !macroA !macroB
SKIP_INSTANCES: macroA_instance1 macroB_instance1
```

In this example, all instances of macroC are skipped and the SKIP_INSTANCES command has no effect.

```
SKIP_CELLS: * macroC
SKIP_INSTANCES: !macroC_instance2
```

Feedback

See Also

- SKIP_CELLS
- INSTANCE_TYPE

Feedback

SKIP_PCELLS

Extracts a parameterized cell (PCell) as a fully characterized gray-box cell unit during parasitic extraction and creates the entity in the DSPF netlist with all layout properties extracted for the PCell.

Syntax

```
SKIP_PCELLS: pcell_name
```

Arguments

Argument	Description
pcell_name	Name of parameterized cell. Only the exclamation mark (!) and asterisk (*) wildcard characters can be used. Default: none

Description

The SKIP_PCELLS command extracts a parameterized cell (PCell) as a fully characterized gray-box cell unit during parasitic extraction and creates the entity in the DSPF netlist with all layout properties extracted for the PCell. The StarRC tool defines a PCell as a container cell, within which one or more devices (including hierarchical cells) are extracted by the ideal device extraction tool. With this command, the tool treats the container cell as a gray-box for parasitic extraction purposes but creates an entry in the DSPF Instance section listing all geometric properties of the ideal device extracted inside the container cell.

In this flow, the PCell placed in layout is assumed to be a fully characterized unit, for which the layout's PCell container cell boundary defines the perimeter between the intradevice effects inside the cell and the interconnect effects outside the cell. Runset terminal layer manipulation is not required to isolate intradevice effects because the PCell cell boundary serves this role. Using the cell boundary eliminates the need to perform runset terminal layer manipulation for PCells while retaining device properties in the netlist. This is the functional benefit of this command.

When you specify the SKIP_PCELLS command, the StarRC tool

- Creates the entity in the DSPF instance section as a device with all layout-extracted device properties; the instantiation name must be consistent with the ideal devices inside the container cell
- Treats the container cell as a gray box (analogous to the SKIP_PCELLS command functionality), which means that parasitic effects are extracted up to the cell boundary

The tool handles exploded shapes in PCells by supporting the following scenarios:

- PCell shapes that are exploded to the upper level
- Flattened designs with some premodeled areas in which the PCell is not preserved

Both scenarios require you to specify the PCell or blocking layer, as well as the layers that are exploded, in a file specified by the `SKIP_PCELL_LAYERS_FILE` command.

The translate.sum file, located in the summary directory, contains the names of cells that are skipped during extraction because they appear in a `SKIP_CELLS` or `SKIP_PCELLS` command.

See Also

- SKIP_CELLS
- SKIP_PCELL_LAYERS_FILE

Feedback

SKIP_PCELL_LAYERS_FILE

Specifies a file that contains information about PCell layers.

Syntax

```
SKIP_PCELL_LAYERS_FILE: file_name
```

Arguments

Argument	Description
file_name	File name Default: none

Description

The SKIP_PCELL_LAYERS_FILE command specifies a file that lists parameterized cell (PCell) layers for the purpose of blocking extraction. The blocked region still contributes to coupling capacitance. You can use the SKIP_PCELL_LAYERS_FILE command for both conducting layers and via layers.

Skipping parameterized cells is a strategy that uses cells in the design to block parasitic extraction from other portions of the design. An alternative strategy uses polygons on a blocking layer to block extraction. The blocking layer approach works with any design hierarchy.

The PCell names must be preserved in the hierarchy of the layout-versus-schematic tool results database, including the Calibre Connectivity Interface or Milkyway XTR view database, although some shapes are exploded.

The file uses the following syntax:

```
PCELL_LAYERS
pcell_name1 layer1 layer2 …
pcell_name2 layer3 layer4 …
pcell_* layer5 layer6 …

BLOCKING_LAYERS
block_layer1 [SIZE svalue | SCALE factor] [TOUCH layerX] layer1 layer2 …
block_layer2 [SIZE svalue | SCALE factor] [TOUCH layerY] layer3 layer4 …
```

To include comments, use a pound sign (#) at the beginning of the line. The StarRC tool ignores all characters on the same line after the pound sign. You cannot begin a comment in the middle of a line.

Requirements for the PCELL_LAYERS section are as follows:

- The first field specifies the PCell name; the asterisk (*) wildcard is allowed. The remaining fields specify the list of exploded layers. Only the found polygons in the blocking layer are retained for use in the blocking layer.
- The PCells must be specified in the `SKIP_PCELLS` command.
- The layers must be specified in the StarRC mapping file.

Requirements for the BLOCKING_LAYERS section are as follows:

- The blocked layer must contain instance pin, port, or probe texts. You cannot use a blocking layer to block parasitics under a marker layer or to block floating nets.
- The first field specifies the blocking layer name.
- The SIZE parameter (in microns) expands or shrinks the blocking layer by the specified amount. A positive value specifies expansion; a negative value indicates shrinkage.
- The SCALE parameter expands or shrinks the blocking layer by the specified factor.
- The (R) flag, when appended to a blocking layer name without any spaces, blocks only resistance extraction on these polygons. The (C) flag appended to the layer name blocks only capacitance extraction. If the layer name appears alone, both resistance and capacitance are blocked. The parentheses are part of the syntax. For example, poly_23(R) blocks only resistance extraction for layer poly_23.
- The TOUCH keyword indicates that a blocking layer polygon is valid only when it touches a polygon in one of the specified TOUCH layers. See the example section for more information.

If you specify the blocking layers section, you do not need to have PCells in the design; the design can be completely flattened. When using this approach,

- The blocking layers must be specified in the Calibre Connectivity Interface or Milkyway XTR view.
- Avoid using both blocking layers and PCells for the same area.

You can specify an ITF layer in both the PCELL_LAYERS and BLOCKING_LAYERS sections. If multiple database layers map to the same ITF layer, specify the ITF layer instead of the database layers at that level. All database layers mapped to the ITF layer are blocking layers. To specify an ITF layer, use the following syntax, with no spaces before or after the double colon:

```
ITF::itfLayerName
```

For example,

```
BLOCKING_LAYERS
SIZE_ID_MN3V3 PWELLC_FINAL SUBS_FINAL ITF::ACTIVE
```

ITF layer names are case-sensitive except when specifying a substrate layer. You can use either the `ITF::SUBSTRATE` or `ITF::substrate` syntax.

Errors

If a layer specified in the BLOCKING_LAYERS section does not block any texts, the StarRC tool issues a warning message.

If the `COUPLE_TO_PCELL_PINS` command is set to `YES`, the StarRC tool cannot calculate the coupling capacitance to the specified PCells, possibly resulting in capacitance underestimation. In such cases, the StarRC tool issues a warning message and provides additional information in the pcelllayers.sum file in the summary directory.

Examples

Use of the PCELL_LAYERS Section

Table 91 illustrates the use of the PCELL_LAYERS section.

Table 91 Example of PCELL_LAYERS Usage

StarRC command to identify PCells	`SKIP_PCELLS: Cell_1 Cell_2`
StarRC command to specify the skip PCell layers file	`SKIP_PCELL_LAYERS_FILE: skip_file1`

Contents of file skip_file1	`PCELL_LAYERS`
	`Cell_1 m1 v1 m2`
	`Cell_3 m3 v3 m4`

Behavior	Cell_1: normal PCell + m1, v1, m2 shapes handlingCell_2: normal PCellCell_3: not considered for PCELL_LAYERS approach

The TOUCH Keyword

The TOUCH keyword provides a way to recognize polygons in layers other than the named blocking layer as blocking polygons. The StarRC tool first finds polygons in the blocking layer that overlap polygons in the touch layer, then uses the overlapped polygons in the touched layer to block the device layer.

The effective region should contain pins (*|I, *|P, or probe text instances). An isolated effective region without any pins might cause incorrect results.

If the TOUCH and SIZE keywords are both used, the StarRC tool first analyzes the relationships defined by the TOUCH keyword, then performs the polygon resizing.

For example, Figure 219 shows two device layers fabricated within an N-well layer. The following line in a blocking layers section applies to this layout:

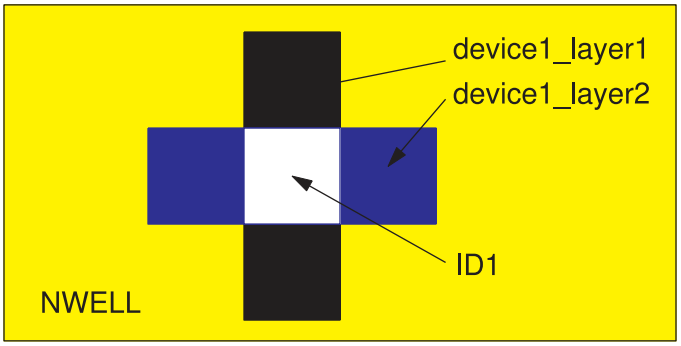
```
NWELL [TOUCH ID1] device1_layer1 device1_layer2
```

Described in terms of design rule checking, the final effective portion of device1_layer1 for RC extraction is as follows (where `not` and `interact` are operations, not layer names):

```
device1_layer1 not (NWELL interact ID1)
```

In other words, extraction for layers device1_layer1 and device1_layer2 is blocked in the region defined by layer ID1, if NWELL completely overlaps the other layers.

Figure 219 Device1 Blocking Example



The following line in a blocking layers section applies to the layout in Figure 220:

```
NWELL [TOUCH a1_blk] dev_layer3
```

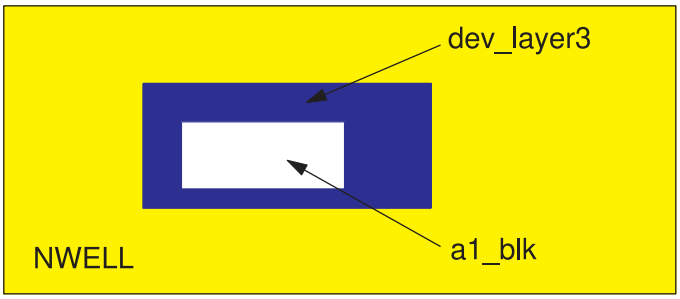
Described in terms of design rule checking, the final effective portion of dev_layer3 for RC extraction is as follows:

```
dev_layer3 not (NWELL interact a1_blk1)
```

In other words, extraction for layer dev_layer3 is blocked in the region defined by layer a1_blk, provided that layer NWELL completely overlaps the other layers.

Feedback

Figure 220 Device2 Blocking Example



See Also

- SKIP_CELLS
- SKIP_PCELLS
- COUPLE_TO_PCELL_PINS

Feedback

SLEEP_TIME_AFTER_FINISH

Specifies the CPU wait time before between runs.

Syntax

```
SLEEP_TIME_AFTER_FINISH: number_of_seconds
```

Arguments

Argument	Description
number_of_seconds	Specifies the CPU wait time Units: seconds Default: 2 Maximum: 2

Description

The SLEEP_TIME_AFTER_FINISH command specifies the CPU wait time between runs.

By default, SLEEP_TIME_AFTER_FINISH is set to 2 seconds. The maximum value allowed is 2 seconds.

Feedback

SMC_AWARE_COUPLING_FILTERING

Filters corners in the simultaneous multicorner (SMC) flow.

Syntax

```
SMC_AWARE_COUPLING_FILTERING: YES | NO
```

Arguments

Argument	Description
YES	Filters corners independently
NO (default)	Default filtering behavior

Description

The default of the SMC_AWARE_COUPLING_FILTERING command is NO. In the SMC flow, by default, the coupling capacitance filtering occurs on the primary corner only and removes the same coupling capacitors from the other corners. The command also preserves the topology between all the corners.

When the SMC_AWARE_COUPLING_FILTERING command is set to YES, the StarRC tool performs filtering on the primary corner and then each of the subsequent SMC corners consecutively, instead of carrying decisions over from the primary corners. Each SMC corner is filtered independently. This means no filtering decision on one corner effects another corner, and the coupling filters are accurate on each corner.

The SMC_AWARE_COUPLING_FILTERING command does not initiate filtering, it only modifies how filtering is done for each corner when filtering is on (SMC_AWARE_COUPLING_FILTERING: YES). There are other commands that control filtering, such as COUPLING_ABS_THRESHOLD, COUPLING_REL_THRESHOLD, and COUPLING_THRESHOLD_OPERATION. These commands also work when the SMC_AWARE_COUPLING_FILTERING is set to NO.

See Also

- COUPLING_ABS_THRESHOLD
- COUPLING_REL_THRESHOLD
- COUPLING_THRESHOLD_OPERATION
- Simultaneous Multicorner Extraction

Feedback

- SIMULTANEOUS_MULTI_CORNER
- SELECTED_CORNERS

Feedback

SMIN_LIMIT

Limits the number of SMIN violations to include in summary reports.

Syntax

```
SMIN_LIMIT: max_count
```

Arguments

Argument	Description
max_count	Maximum number of SMIN violations to report Default: 1000

Description

In cases where the StarRC tool identifies a large number of SMIN violations, you might want to limit the amount of detail included in summary reports and the number of times that similar messages are issued.

The `SMIN_LIMIT` command specifies the maximum number of unique SMIN violations for which to report detailed information such as layer names and bounding boxes in the smin.sum file. If the limit is exceeded, the StarRC tool writes a warning message in the file and does not report the additional violations.

By default, the StarRC tool does not report SMIN violations. To enable reporting, set the `REPORT_SMIN_VIOLATION` command to `YES`.

See Also

- REPORT_SMIN_VIOLATION
- SMIN Violations Reports

Feedback

SNAP_RESISTOR_WIDTH

Changes the widths of design resistors that are less than the layer WMIN value to be equal to the WMIN value and generates a report.

Syntax

```
SNAP_RESISTOR_WIDTH: YES | NO
```

Arguments

Argument	Description
YES (default)	If a resistor width is less than the layer WMIN value, changes the width to the WMIN value
NO	Leaves resistor widths unchanged

Description

The StarRC tool always uses actual design widths during resistance extraction. However, when the SNAP_RESISTOR_WIDTH command is set to YES (the default), polygons that are narrower than the layer WMIN value are set to have a width equal to the WMIN value in the output netlist.

When resistor snapping is enabled, the StarRC tool generates a report named snap_width_report that lists the nodes of all resistances whose widths are snapped.

For example, consider a resistance whose drawn width is 0.4 microns on a layer with a WMIN value of 0.5. If width snapping is enabled, the SPEF netlist output might be as follows:

```
3 *421:A *330:6 1.10000 // $l=0.600000 $w=0.500000 $lvl=1
```

The snap_width_report file then contains the following lines:

```
From_Node To_Node
*421:A *330:6
```

To snap the resistor widths, the NETLIST_TAIL_COMMENTS command must be set to YES and the DETECT_FUSE command must be set to NO (the default). Table 92 shows the effects of these command settings on resistor width.

Table 92 Effect of Command Settings on Resistor Width Snapping

SNAP_RESISTOR_WIDTH	NETLIST_TAIL_COMMENTS	DETECT_FUSE	Snap resistor width?
NO	YES	NO	Yes (tool sets SNAP_RESISTOR_WIDTH command to YES)
NO or unset	YES	YES	No
NO or unset	NO	YES	No
NO or unset	NO	NO	No
YES	YES	NO	Yes
YES	YES	YES	No (tool sets SNAP_RESISTOR_WIDTH command to NO and issues a warning)
YES	NO	n/a	No (tool sets SNAP_RESISTOR_WIDTH command to NO and issues a warning)

See Also

- WMIN
- NETLIST_TAIL_COMMENTS
- DETECT_FUSE

Feedback

SPF_CHECKS

Parasitic netlist checker that operates on SPF to verify the output of a netlist in a transistor-level flow.

Syntax

```
SPF_CHECKS
    [-output output_file_directory]
    [-skip_nets list_of_nets]
    [-cap_rel relative_cap_error_threshold]
    [-resistor_threshold threshold_value]
    [-node_cap_threshold_abs absolute_cap_threshold]
    [-node_cap_threshold_rel relative_cap_threshold]
    [-node_cap_threshold_operation AND | OR]
    [-ignore_ln_ports yes | no]
    [-ignore_ln_nets yes | no]
    [-rename_file YES | NO]
```

Description

For the following information, see Parasitic Netlist Checker:

- Options and arguments with descriptions
- Checks performed by each option and the name of the output file generated or error message issued by the tool

Note:

Use the `SPF_CHECKS` command only in the StarRC command file as shown in Example 38

Example 38 Using the SPF_CHECKS Command in the StarRC Command File

```
SPF_CHECKS: -skip_nets VSS,VDD -node_cap_threshold_operation and \
    -rename_file YES -ignore_ln_ports No -ignore_ln_nets yes \
    -cap_rel 0.01 -resistor_threshold 10 -node_cap_threshold_abs 0.5 \
    -node_cap_threshold_rel 0.3
```

Feedback

SPICE_SUBCKT_FILE

Specifies one or more SPICE files that contain .SUBCKT definitions for skip cells.

Syntax

```
SPICE_SUBCKT_FILE: file1 [… fileN]
```

Arguments

Argument	Description
file1 [… *fileN*]	Files containing the subcircuit definitions for all skip cells in the design Default: none

Description

The StarRC tool reads the files specified by the `SPICE_SUBCKT_FILE` command to obtain port ordering information. The files also control the port ordering of the top cell. The port order and the port list members read from the .SUBCKT definition for a skip cell are preserved in the output netlist.

The following usage notes apply:

- This command and the specified files do not control the port ordering for the devices.
- SPICE `.include` statements are not supported and should not appear in the specified files.
- Only the SPF, NETNAME, and STAR netlist formats are supported.

See Also

- NETLIST_IDEAL_SPICE_FILE
- SKIP_CELLS

Feedback

STAR_DIRECTORY

Sets the star directory name.

Syntax

```
STAR_DIRECTORY: directory
```

Arguments

Argument	Description
directory	A valid directory name Default: star

Description

The STAR_DIRECTORY command specifies a directory that the StarRC tool creates for reports and intermediate files. The command defaults to the string "star." As a result, StarRC documentation often uses the term "star directory" as a generic term. However, you can use any string that follows valid directory naming conventions.

The STAR_DIRECTORY command accepts both absolute and relative paths. However, you can specify a relative path only if the directory is below the working directory (the working directory is the directory from which you run the StarXtract command). For example:

```
% star_dir/working_dir/other_dir     (incorrect)
% working_dir/other_dir/star_dir     (correct)
```

For use with simultaneous multicorner extraction, the arguments in the CORNERS_FILE and STAR_DIRECTORY commands must follow these naming conventions:

- If the STAR_DIRECTORY command argument is a relative path, you can use either a relative path or an absolute path in the CORNERS_FILE command. For example:

  ```
  STAR_DIRECTORY: star
  CORNERS_FILE: smc_config
  ```

- If the STAR_DIRECTORY command argument is an absolute path, you must use an absolute path in the CORNERS_FILE command. For example:

  ```
  STAR_DIRECTORY: /tmp/star
  CORNERS_FILE: /remote/.../work_directory/smc_config
  ```

See Also

- CORNERS_FILE
- SIMULTANEOUS_MULTI_CORNER

Feedback

STARRC_DP_MIN_CORES

Specifies the minimum number of cores to be available to proceed with a distributed processing run.

Syntax

```
STARRC_DP_MIN_CORES: core_count
STARRC_DP_MIN_CORES: core_pct%
```

Arguments

Argument	Description
core_count	Minimum number of cores, expressed as an integer Default: 1 Minimum: 1 Maximum: The value of the NUM_CORES command
core_pct%	Minimum percentage of cores . The percent symbol (%) is part of the syntax and must be included. Default: n/a (1 core)

Description

The STARRC_DP_TIME_OUT and STARRC_DP_MIN_CORES commands work together to define the computing resources necessary to proceed with a distributed processing run and the maximum amount of time to wait for those resources to become available.

Use the STARRC_DP_MIN_CORES command to specify the minimum number of cores necessary to proceed with the run. You can set the value in either of the following ways:

- Specify an integer number of cores.
- Specify a percentage of the value of the NUM_CORES command:

  Minimum cores = NUM_CORES * *core_pct*/100

Note:

If you use the STARRC_MIN_CORES command, you must also use the STARRC_DP_TIME_OUT command or the run never begins. However, you can use the STARRC_DP_TIME_OUT command alone, in which case the run proceeds when one core becomes available.

Feedback

See Also

- STARRC_DP_STRING
- STARRC_DP_TIME_OUT

Feedback

STARRC_DP_STRING

Enables automatic submission of distributed processing jobs.

Syntax

```
STARRC_DP_STRING:
      bsub lsf_arguments
    | qsub gridware_arguments
    | list [login_protocol] host1[:n1] [host2[:n2] ... hostm[:nm]]
    | list localhost:num_processes
    | nc run rtda_arguments
    | sbatch -p partition_name
```

Arguments

Argument	Description
lsf_arguments	Arguments for an LSF system
gridware_arguments	Arguments for a Gridware system
login_protocol	Login protocol. Valid values: `rsh` (default) or `ssh` Using the ssh protocol requires additional setup. For more information, see Distributed Processing.
host1, host2 ...	Name of host machine
n1, n2 ...	Number of runs to submit on corresponding host
num_processes	Number of processes on the local host
rtda_arguments	Arguments for a Runtime Design Automation (RTDA) system
`sbatch`	Executes jobs in the default partition queue In the StarRC command file, specify the `STARRC_DP_STRING: sbatch` command only. To use the `-P` option with `sbatch`, see Example 39.

Description

Distributed processing allows you to start a single StarRC run and let the tool automatically submit multiple jobs. You can specify the job submission command by setting the `STARRC_DP_STRING` environment variable or by using the `STARRC_DP_STRING` command in the StarRC command file. If both the environment variable and command are set, the `STARRC_DP_STRING` command in the command file takes precedence.

The control parameters in the submission command are site-specific; contact your system administrator for assistance.

Feedback

Distributed processing is available for the following computing environments:

- A single host
- A general network with a list of machines
- LSF system
- Gridware system
- Runtime Design Automation (RTDA) system

For more information, see Distributed Processing.

Note:

The number of worker processes launched by the StarRC tool is equal to the setting of the `NUM_CORES` command. If your submission command specifies a larger number of cores, some cores are reserved but not used. For best results, the StarRC `NUM_CORES` command and the submission command should specify the same number of cores.

Examples

On an LSF system:

```
STARRC_DP_STRING: bsub -R "rusage[mem=5000]"
```

On a Gridware system:

```
STARRC_DP_STRING: qsub -P bnormal -l "mem_free=1G mem_avail=1G"
```

This example for a general network uses the ssh protocol and submits 4 runs on system alpha, 2 runs on system beta, and 1 run on system gamma:

```
STARRC_DP_STRING: list ssh alpha:4 beta:2 gamma
```

On an RTDA system:

```
STARRC_DP_STRING: nc run -P bnormal -l "mem_free=1G mem_avail=1G"
```

On a Slurm Workload Manager:

Example 39 Requesting a specific partition queue to allocate resources

```
STARRC_DP_STRING: sbatch -P debug -N 2 --mem-per-cpu=1G
```

See Also

- ENABLE_IPV6
- NUM_CORES

Feedback

- STARRC_DP_MIN_CORES
- STARRC_DP_TIME_OUT
- Distributed Processing

Feedback

STARRC_DP_STRING_FOR_HIGH_MEM_TASKS

Enables automatic distributed processing of high memory tasks with high memory hosts.

Syntax

```
STARRC_DP_STRING_FOR_HIGH_MEM_TASKS:
      bsub lsf_arguments
    | qsub gridware_arguments
    | list [login_protocol] host1[:n1] [host2[:n2] ... hostm[:nm]]
    | list localhost:num_processes
    | nc run rtda_arguments
    | sbatch -p partition_name
```

Arguments

Argument	Description
lsf_arguments	Arguments for an LSF system
gridware_arguments	Arguments for a Gridware system
login_protocol	Login protocol. Valid values: `rsh` (default) or `ssh` Using the ssh protocol requires additional setup. For more information, see Distributed Processing.
host1, host2 ...	Name of the host machine
n1, n2 ...	Number of runs to submit on the corresponding host
num_processes	Number of processes on the local host
rtda_arguments	Arguments for a Runtime Design Automation (RTDA) system
`sbatch`	Executes jobs in the default partition queue In the StarRC command file, specify the `STARRC_DP_STRING: sbatch` command only.

Description

Distributed processing allows you to start a single StarRC run to automatically submit multiple jobs.

The settings are same as the settings of the `STARRC_DP_STRING` command. You must specify the same distributed processing protocols such as (LSF, UGE, and SLURM) with the `STARRC_DP_STRING_FOR_HIGH_MEM_TASKS` and `STARRC_DP_STRING` commands. However, if you specify different distributed protocols with the commands, the StarRC tool issues an error message.

Feedback

See Also

- Distributed Processing
- STARRC_DP_STRING

Feedback

STARRC_DP_TIME_OUT

Specifies the maximum time to wait for cores to become available.

Syntax

```
STARRC_DP_TIME_OUT: timeout
```

Arguments

Argument	Description
max_count	Maximum time to wait for cores to become available Units: minutes Default: 0 (no timeout)

Description

The STARRC_DP_TIME_OUT and STARRC_MIN_CORES commands work together to define the computing resources necessary to proceed with a distributed processing run and the maximum amount of time to wait for those resources to become available.

If the STARRC_DP_TIME_OUT command is set to zero (the default) or a negative value, the wait time is infinite.

Note:

If you use the STARRC_MIN_CORES command, you must also use the STARRC_DP_TIME_OUT command or the run never begins. However, you can use the STARRC_DP_TIME_OUT command alone, in which case the run proceeds when one core becomes available.

See Also

- STARRC_DP_MIN_CORES
- STARRC_DP_STRING

Feedback

STOP_EXTRACTION_ON_NUMEROUS_SHORTS

Specifies whether to stop extraction if a large number of shorts is detected.

Syntax

```
STOP_EXTRACTION_ON_NUMEROUS_SHORTS: YES | NO
```

Arguments

Argument	Description
YES (default)	Stops extraction if too many shorts are detected
NO	Continues extraction even when many shorts are detected

Description

The STOP_EXTRACTION_ON_NUMEROUS_SHORTS command allows you to choose whether to continue extraction in the presence of shorts.

Runtime can be very long for a design that contains a large number of shorts. This condition might result from an unknown problem in the design data, in which case the appropriate course of action is to stop the run and resolve the problem.

In other cases, extraction data might be run on an incomplete design, in which case the shorts are expected and can be ignored in the output data. In this case, the appropriate course of action is to continue processing despite the presence of shorts.

By default, execution stops if both of the following conditions are true:

- The number of shorts is greater than the number of polygons in the partition.
- The number of shorts is greater than 25000.

Set the STOP_EXTRACTION_ON_NUMEROUS_SHORTS command to NO to continue execution.

Feedback

SUBSTRATE_EXTRACTION

Extracts resistance from layers mapped to substrate in the mapping file.

Syntax

```
SUBSTRATE_EXTRACTION: YES | NO
```

Arguments

Argument	Description
YES	Performs substrate extraction on layers designated as substrate layers in a conducting_layers section of the mapping file
NO (default)	Does not perform substrate extraction

Description

The SUBSTRATE_EXTRACTION command enables substrate extraction. You must also set the TRANSLATE_RETAIN_BULK_LAYERS command to a value different from the default of NO in the StarRC command file.

You must specify each substrate layer in the mapping file by using the keyword substrate in a conducting_layers section. At least one substrate layer must be defined in the mapping file. You can include RPSQ values for each layer, and the values can be different for each layer. Substrate layers without RPSQ values are treated as ideal.

Since bulk layers are large and highly resistive, the StarRC tool performs mesh extraction for these layers, resulting in an increased parasitic netlist size.

Use the SUBSTRATE_EXTRACTION command to obtain substrate resistance information for the following purposes:

- To prevent shorting of multiple electrical nodes that exist within a common substrate well for the purpose of analyzing power nets having multiple electrical taps to a common well
- For parasitic viewing applications, to enable resistive probing between nodes that share a common well
- To allow the extraction of resistance between a the bulk terminal and the source and drain terminals of a MOS device for the purpose of simulating back-biasing effects

This feature is not intended to facilitate substrate extraction for purposes of substrate noise modeling.

Feedback

Examples

The following example shows a mapping file:

```
(with substrate extraction)
conducting_layers
   nwell     SUBSTRATE RPSQ=1000
   psub      SUBSTRATE RPSQ=2000

(with no substrate extraction)
conducting layers
   nwell     SUBSTRATE
   psub      SUBSTRATE
```

See Also

- RPSQ
- TRANSLATE_RETAIN_BULK_LAYERS

Feedback

SUMMARY_FILE

Specifies the name of the summary file.

Syntax

```
SUMMARY_FILE: file_name
```

Arguments

Argument	Description
file_name	The name of the summary file Default: *block_name*.star_sum

Description

The SUMMARY_FILE command specifies the name of the summary file.

By default, the summary file is located in the run directory and has the name *block_name*.star_sum, where *block_name* is the block specified by the BLOCK command.

You can use the SUMMARY_FILE command to change the name and location of the summary file. This command accepts a path relative to the run directory. However, an absolute path is not permitted because of the possibility of a change in the network file system.

Examples

To create a summary file my_summary.log in the results subdirectory, use the following syntax in the StarRC command file:

```
SUMMARY_FILE: ./results/my_summary.log
```

See Also

- BLOCK

Feedback

SUPPORT_DIFFERENT_PORTNAME_NETNAME

Specifies whether to allow different names for ports and the nets connected to those ports.

Syntax

```
SUPPORT_DIFFERENT_PORTNAME_NETNAME: YES | NO
```

Arguments

Argument	Description
YES	Supports different port names and net names
NO (default)	Does not support a different port name on a net. Different port names are dropped.

Description

By default, the StarRC tool assumes that a net connected to a port has the same name as the port. If the names are different, the tool issues a warning message, keeps the net, and does not include the port in the netlist.

For the Calibre Connectivity Interface transistor-level flow, you can set the SUPPORT_DIFFERENT_PORTNAME_NETNAME command to YES to allow ports and the nets connected to them to have different names. In addition, one net can connect to multiple ports, all with different names.

If the NETLIST_IDEAL_SPICE_FILE command is used to create an ideal SPICE file, the new ports are included in the .SUBCKT definitions.

Table 93 shows the effects of the command setting.

Table 93 Different Port Name and Net Name Behavior

Port name	Net name	Behavior if NO (default)	Behavior if YES
VSS	VSS	.SUBCKT BLOCK VSS ... *|NET VSS *|P VSS	.SUBCKT BLOCK VSS ... *|NET VSS *|P VSS
VSS	VSS_1	.SUBCKT BLOCK VSS_1 ... *|NET VSS_1 (no *|P created; warning issued)	.SUBCKT BLOCK VSS ... *|NET VSS_1 *|P VSS
VSSA VSSB VSSC	VSS_1	.SUBCKT BLOCK VSS_1 ... *|NET VSS_1 (no *|P created; warning issued)	.SUBCKT BLOCK VSSA VSSB VSSC... *|NET VSS_1 *|P (VSSA ...) *|P (VSSB ...) *|P (VSSC ...)

A different port name overrides port names specified in the `SPICE_SUBCKT_FILE` and `PIO_FILE` commands.

See Also

- NETLIST_IDEAL_SPICE_FILE
- SPICE_SUBCKT_FILE
- PIO_FILE

Feedback

TARGET_PWRA

Generates the StarRC commands needed for power reliability analysis. Valid only for transistor-level flows.

Syntax

```
TARGET_PWRA: YES | NO
```

Arguments

Argument	Description
YES	Generates an optimized set of commands for reliability analysis using the StarRC or HSIM flow
NO (default)	Does not generate the reliability analysis commands

Description

The TARGET_PWRA command automatically generates StarRC commands for RC extraction in a power reliability analysis flow. With this command, you must also use the POWER_NETS command to specify a list of power nets.

The TARGET_PWRA: YES command causes the StarRC tool to generate two netlists. One netlist contains unreduced resistors for power nets. The second netlist contains reduced RC-coupled devices for signal nets. The signal netlist is useful for both reliability analysis and signal timing analysis.

The TARGET_PWRA: YES command generates the following commands and overrides any commands of the same type that appear elsewhere in the command file:

- POWER_EXTRACT: RONLY

  Creates an additional resistor-only netlist when the NETLIST_POWER_FILE and POWER_EXTRACT: YES commands are used.

- POWER_REDUCTION: LAYER_NO_EXTRA_LOOPS

  Specifies that reduction is applied to the specified power nets rather than signal nets.

  This option is specified if no other instance of the POWER_REDUCTION command appears in the command file or if you set the POWER_REDUCTION command to YES elsewhere in the command file. However, if you set the POWER_REDUCTION command to the more conservative NO or LAYER options, those settings are not changed.

- NETLIST_CONNECT_SECTION: YES

  Generates *|I, *|P, and *CONN sections in the output file.

Feedback

- NETLIST_NODE_SECTION: YES

  Generates *|S and *N statements in the output file.

- EXTRA_GEOMETRY_INFO: RES NODE

  Writes node and resistor geometry information as comments in the netlist.

- NETLIST_TAIL_COMMENTS: YES

  Enables writing the extra geometry information in the netlist.

- NETLIST_FORMAT: SPF

  Specifies the SPF netlist format.

- NETLIST_POWER_FILE: *block_name.pwr.spf*

  Specifies the name for the output netlist that contains power rail resistor values.

- SHORT_PINS: NO

  Specifies not to short top-level pin ports that have multiple placements.

You do not need to set the EXTRA_GEOMETRY_INFO: NODE command because *|S is the center of the bounding box node.

If the TARGET_PWRA: YES command is used and the MAX_VIA_ARRAY_SPACING option of the via_layers mapping file command is not set, the StarRC tool sets the value of the MAX_VIA_ARRAY_SPACING option as follows:

- If the AREA option of the via_layers mapping file is set, the MAX_VIA_ARRAY_SPACING parameter value is set to the square root of the AREA option value.
- If the AREA option is not set but an RPV_VS_AREA table exists in the VIA statement, the MAX_VIA_ARRAY_SPACING parameter is set to the square root of the first area value in the RPV_VS_AREA table.
- AREA values specified in a mapping file take precedence over RPV_VS_AREA values specified in an ITF file.

Examples

```
BLOCK: blockname
MILKYWAY_DATABASE: mw_file
MILKYWAY_EXTRACT_VIEW
TARGET_PWRA:YES
POWER_NETS: list_of_power_nets
TCAD_GRD_FILE: nxt_file
MAPPING_FILE:  map_file
XREF: YES
SKIP_CELLS: list_of_cells
NETLIST_FILE: blockname_signal_nets.spf
```

Feedback

See Also

- EXTRA_GEOMETRY_INFO : NODE
- KEEP_VIA_NODES : NO
- via_layers
- NETLIST_CONNECT_SECTION
- NETLIST_FORMAT
- NETLIST_POWER_FILE
- NETLIST_NODE_SECTION
- NETLIST_TAIL_COMMENTS
- POWER_EXTRACT
- POWER_NETS
- POWER_REDUCTION
- REDUCTION
- SHORT_PINS

Feedback

TCAD_GRD_FILE

Specifies the name of the nxtgrd file, which contains process model information.

Syntax

```
TCAD_GRD_FILE: nxtgrd_file1
TCAD_GRD_FILE: nxtgrd_file1 nxtgrd_file2 nxtgrd_file3
```

Arguments

Argument	Description
nxtgrd_file1, nxtgrd_file2, ...	Name of the nxtgrd file or files Default: none

Description

The TCAD_GRD_FILE command specifies the nxtgrd file, which contains process information such as capacitance models and conductor resistances. This command is mandatory for all extraction flows.

You can create an nxtgrd file using the grdgenxo tool on a file that contains ITF process modeling commands. However, for signoff flows, you must use an nxtgrd file obtained from your foundry.

Using the nxtgrd file requires information that maps every process layer in the nxtgrd file to the corresponding layout database layer. For gate-level flows, specify the mapping file with the MAPPING_FILE command. For transistor-level flows, you can optionally specify a runset report file from the LVS tool instead of a mapping file.

See Also

- MAPPING_FILE
- Flows for Process Characterization
- The Mapping File

Feedback

TEMPERATURE_SENSITIVITY

Enables the temperature sensitivity flow for transistor-level extraction.

Syntax

```
TEMPERATURE_SENSITIVITY: YES | NO
```

Arguments

Argument	Description
YES	Enables temperature sensitivity analysis
NO (default)	Disables temperature sensitivity analysis

Description

The TEMPERATURE_SENSITIVITY: YES command writes the parasitic resistor temperature coefficients TC1 and TC2 to the netlist for use by simulation tools. TC1 and TC2 are obtained from CRT1 and CRT2 in the ITF file. All reduction modes are supported.

Temperature coefficients are not reported if they are not set or if they are equal to 0. If the absolute value of TC1 is less than 1e-06, it is set to 1e-06. If the absolute value of TC2 is less than 1e-09, it is set to 1e-09.

If the value of TC2 is so small that ignoring it does not significantly affect the resistance error, TC2 might not be written to the netlist.

To report temperature coefficients in SPF, STAR, and NETNAME netlists, the StarRC tool creates a SPICE model card named resStar for parasitic resistors. This model card is placed inside the .subckt definition for the top-level cell. In the following example, the top cell name is dummy_ABC and the cell has two ports:

```
.subckt dummy_ABC port1 port2
.model resStar R Tref=25
...
.ends
```

The format of the coefficients depends on the netlist type, as follows:

- In the resStar model, temperature coefficients are labeled TC1 and TC2, as follows:

```
R1 n1:1 n2:2 resStar R=78 TC1=0.0012556 TC2=-1.95301e-07
```

- In a SPEF netlist, temperature coefficients are written using the sensitivity format. The field definitions are written in the *VARIATION_PARAMETERS section of the netlist. The index number of the TC1 and TC2 values might vary in different netlists.

```
*VARIATION_PARAMETERS
6 CRT1
7 CRT2 25.0000
*RES
1 p1:A p3:Z 2.50093 *SC 4:0.900 5:0.531 6:0.00321 7:-.000021
```

The `TEMPERATURE_SENSITIVITY: YES` command can be used with the simultaneous multicorner flow. The following conditions apply:

- The supported temperature range is -55 C to 150 C.
- The maximum number of selected process corners is 15.
- The `OPERATING_TEMPERATURE` settings in the corners file are ignored.
- If the selected corners contain redundant nxtgrd files, the tool discards the redundant corners, determines the temperature coefficients for the remaining corners, and issues a warning message.

See Also

- Writing Resistor Temperature Coefficients to the Netlist

Feedback

TOP_DEF_FILE

Specifies the top-level block design file in DEF format.

Syntax

```
TOP_DEF_FILE: def_file
```

Arguments

Argument	Description
def_file	The top block design file in DEF format Default: none

Description

This command defines the top-level block for extraction and is mandatory for LEF/DEF flows.

This DEF file can reference macros defined in separate files with the `MACRO_DEF_FILE` command. The standard cell and routing layer definitions should be defined in the accompanying LEF file. Macro blocks appearing in the DEF file specified by the `TOP_DEF_FILE` command are skipped by default.

You can specify gzip files with the `TOP_DEF_FILE` command.

See Also

- LEF_FILE
- MACRO_DEF_FILE

Feedback

TRANSLATE_DEF_BLOCKAGE

Translates the routing blockages from a top-level DEF file.

Syntax

```
TRANSLATE_DEF_BLOCKAGE: YES | NO | SHRINK
```

Arguments

Argument	Description
YES	Translates the routing blockages from a designated top DEF file
NO (default)	Does not translate DEF file blockages
SHRINK	Translates BLOCKAGE objects with shrink spacing as real metal shapes and allows coupling to these shapes

Description

This command translates the routing blockages from a top-level DEF file (specified in the TOP_DEF_FILE command). Blockages from files specified by the MACRO_DEF_FILE command are ignored because the routing information corresponding to those blockages is already present in the top-level DEF file.

Placement blockages in the top-level DEF file are ignored.

Examples

This example shows the BLOCKAGES section of a DEF file.

```
[BLOCKAGES numBlockages ;
   [- LAYER layerName
      [+ COMPONENT compName | + SLOTS | + FILLS | + PUSHDOWN]
      [+ SPACING minSpacing | + DESIGNRULEWIDTH effectiveWidth]
         {RECT pt pt | POLYGON pt pt pt …} …
;] …
[- PLACEMENT
   [+ COMPONENT compName | + PUSHDOWN]
   {RECT pt pt} …
;] …
```

See Also

- MACRO_DEF_FILE
- TOP_DEF_FILE

Feedback

TRANSLATE_DEF_BLOCKAGE_TYPE

Translates DEF layer blockages of specific types as real metal shapes.

Syntax

```
TRANSLATE_DEF_BLOCKAGE_TYPE: ALL | FILLS | SLOTS | PUSHDOWN | LAYER
```

Arguments

Argument	Description
ALL	Translates all types of DEF layer blockages.
FILLS	Translates only the fills type DEF layer blockages. The FILLS and SLOTS arguments are mutually exclusive.
SLOTS	Translates only the slots type DEF layer blockages. The SLOTS and FILLS arguments are mutually exclusive.
PUSHDOWN	Translates only the pushdown type DEF layer blockages.
LAYER	Translates only the DEF layer blockages that are set with the EXCEPTPGNET, DESIGNRULEWIDTH, or SPACING property.

Description

This command translates all DEF layer blockages from a top-level DEF file to allow coupling to these shapes when the TRANSLATE_DEF_BLOCKAGE command is set to YES.

If you do not use the TRANSLATE_DEF_BLOCKAGE_TYPE command, the tool translates all layer blockages.

See Also

- TRANSLATE_DEF_BLOCKAGE

Feedback

TRANSLATE_DRCFILL_AS_OBS

Translates DRCFILL polygons in the SPECIALNETS section of a DEF file as if they were OBS polygons.

Syntax

```
TRANSLATE_DRCFILL_AS_OBS: YES | NO
```

Arguments

Argument	Description
`YES` (default)	Translates DRCFILL polygons in the SPECIALNETS section of a DEF file
`NO`	Does not translate DRCFILL polygons

Description

Designs for double patterning processes might contain extra polygons to enable the advanced lithography. For designs created in the NDM format by the Fusion Compiler or IC Compiler II tool, the extension polygons are marked internally according to whether they belong to normal nets, unconnected nets, or obstruction (OBS) geometries.

The StarRC tool can read and translate all such polygons when the Fusion Compiler or IC Compiler II logic library is read directly.

However, if a logic library from the Fusion Compiler or IC Compiler II tool is saved in a LEF/DEF format, the polygons from the normal nets are written in the NETS section of the DEF file while the polygons from unconnected or OBS nets are written in the SPECIALNETS section. Using the LEF/DEF version of the design as input to the StarRC tool requires that the extension polygons in the SPECIALNETS section be recognized.

The `TRANSLATE_DRCFILL_AS_OBS` command instructs the StarRC tool to translate all of the DRCFILL polygons in a DEF file as if they were OBS polygons.

See Also

- NDM_DATABASE

Feedback

TRANSLATE_FLOATING_AS_FILL

Considers disconnected floating polygons as fill polygons or connected to ground.

Syntax

```
TRANSLATE_FLOATING_AS_FILL: YES | NO
```

Arguments

Argument	Description
YES	Treats disconnected floating polygons as fill polygons. Their capacitive interaction is accounted as metal fill polygons.
NO (default)	Treats disconnected floating polygons as simple ideal ground

Description

The StarRC tool handles floating and grounded metal fill through a separate metal fill GDSII file interface for transistor-level flows. For these flows, the name of a net is based on pin-marker definitions from layout versus schematic (LVS) tools. For nets that do not have pin-marker layers or text, LVS tools generally assign a random net ID to these layers. These polygons are considered disconnected or floating. Since these polygons are present in the input database, the StarRC tool must take the polygons into account for capacitive interaction. Resistance is not a concern because there are no pins present and thus no current flowing through these polygons.

The following usage notes apply:

- The TRANSLATE_FLOATING_AS_FILL command is independent of the METAL_FILL_POLYGON_HANDLING command.
- The TRANSLATE_FLOATING_AS_FILL: YES command takes precedence over the REMOVE_FLOATING_NETS: YES command.

See Also

- METAL_FILL_POLYGON_HANDLING
- REMOVE_FLOATING_NETS

Feedback

TRANSLATE_NDM_BLOCKAGE

Translates the routing blockages from an IC Compiler II or a Fusion Compiler design library.

Syntax

```
TRANSLATE_NDM_BLOCKAGE: YES | NO
```

Arguments

Argument	Description
YES	Enables translation of the routing blockages from a designated design
NO (default)	Disables translation of routing blockages

Description

The TRANSLATE_NDM_BLOCKAGE command enables the translation of the routing blockages from an IC Compiler II or a Fusion Compiler design library.

See Also

- NDM_DATABASE
- NDM_DESIGN_VIEW

Feedback

TRANSLATE_RETAIN_BULK_LAYERS

Specifies how to treat mapped bulk layers during transistor-level extraction.

Syntax

```
TRANSLATE_RETAIN_BULK_LAYERS: YES | CONLY BULK_THRU_WELLTAPS
 [ALL_TO_CLOSEST_CONTACT]
```

Arguments

Argument	Description
YES	Passes all mapped bulk layers for extraction
NO (default)	Discards bulk layers
CONLY	Passes all mapped bulk layers for capacitance-only extraction. Connects device bulk terminals ideally to the port with the same name as the connected net name in the instance section of the netlist, but does not report the terminals in the detailed NET section. Also connects nonbulk terminals on SUBSTRATE layers (such as terminals of NWELL resistors or ESD protection diodes) to the closest substrate contacts ideally.
CONLY ALL_TO_CLOSEST_CONTACT	Passes all mapped bulk layers for capacitance-only extraction, connected to the closest tap. Bulk connections are written as *|I instances, allowing back-annotation to bulk nodes.
CONLY BULK_THRU_WELLTAPS	Retains all well tap polygons for extraction but does not short the well tap polygons through SUBSTRATE layers. Also, the tool does not preserve the bulk connection, resulting in the bulk terminal getting connected to ideal nets.
CONLY BULK_THRU_WELLTAPS ALL_TO_CLOSEST_CONTACT	The behavior is similar to CONLY ALL_TO_CLOSEST_CONTACT. In addition, this option retains bulk terminals, as well as all well tap polygons for extraction but does not short the well tap polygons through SUBSTRATE layers. As a result, the bulk connection is reported as a *|I instance in the netlist file, allowing back-annotation on bulk nodes by simulation tools.

Description

The TRANSLATE_RETAIN_BULK_LAYERS command specifies how the StarRC tool handles bulk layers during transistor-level extraction. This command has an effect only if the bulk layers are mapped in the mapping file.

A bulk layer is defined as any database layer that is used as the bulk terminal layer for any of the following devices: NMOS, PMOS, resistor, diode, or bipolar junction transistor. A nonbulk layer is any other layer that is physically part of the substrate, such as a well.

Note:

In Calibre Connectivity Interface flows, bulk layer recognition applies to devices that have a device type of MOS in the CALIBRE_DEVTAB file (including MN, MP, MD, and ME element names) and also to devices that have the element name 'M'. Bulk layer recognition does not apply to lightly-doped-drain (LDD) devices, because those source and drain pins are not swappable, or to most devices bearing the USER tag in the DEVICE_TYPE field of the CALIBRE_DEVTAB file, including inductor devices.

For accurate capacitance extraction, specify the layer precedence in the mapping file. Specifying the precedence is required if more than one database layer is mapped to the substrate and the `TRANSLATE_RETAIN_BULK_LAYERS` command is set to any value other than `NO`.

The YES Option

The StarRC tool extracts the coupling capacitance to device bulk layers that serve as device terminal layers. In this mode, the tool outputs an instance port subnode for the bulk terminal of the devices in the detailed NET section of the netlist.

The NO Option (default)

The StarRC tool includes the capacitance to device bulk layers as a generic ground capacitance in the netlist. In this mode, the tool connects device bulk terminals ideally to the port object with the same name as the connected net name in the instance section of the netlist, but does not report the terminals in the detailed NET section.

The CONLY Option

The tool extracts the coupling capacitance to device bulk layers that serve as device terminal layers. In this mode, the tool connects device bulk terminals ideally to the port object with the same name as the connected net name in the instance section of the netlist output, but does not report the terminals in the detailed NET section of the netlist.

In addition, device nonbulk terminals on `SUBSTRATE` layers are connected to the closest substrate contacts ideally with a shorting resistor. The tool outputs an instance port subnode for these nonbulk terminals in the detailed NET section of the netlist.

If a net does not have a top-level pin (a *|P entry), the bulk terminals are connected to the closest contact.

If you want to retain all of the substrate contacts (instead of the default of one contact) for certain models when using the `TRANSLATE_RETAIN_BULK_LAYERS:CONLY` command, use the `KEEP_SUBCONT_MODELS` command to specify the models of interest.

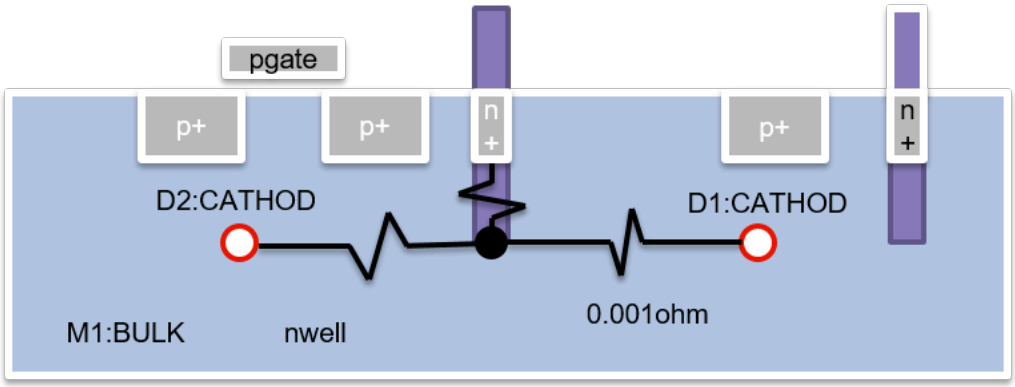
Feedback

Note:

If the SHORT_PINS command is set to YES (the default), an additional virtual connection is assumed to exist between multiple pins that have the same text. Different groups of substrate contacts might have a connection path through the virtual connection. Using the TRANSLATE_RETAIN_BULK_LAYERS:CONLY command might cause different results depending on whether the SHORT_PINS command is set to YES or NO, due to the processing needed to avoid shorting effects.

Figure 221 illustrates the TRANSLATE_RETAIN_BULK_LAYERS:CONLY option. The bulk layers are extracted for capacitance and connected to an ideal net.

Figure 221 Effect of the CONLY Option



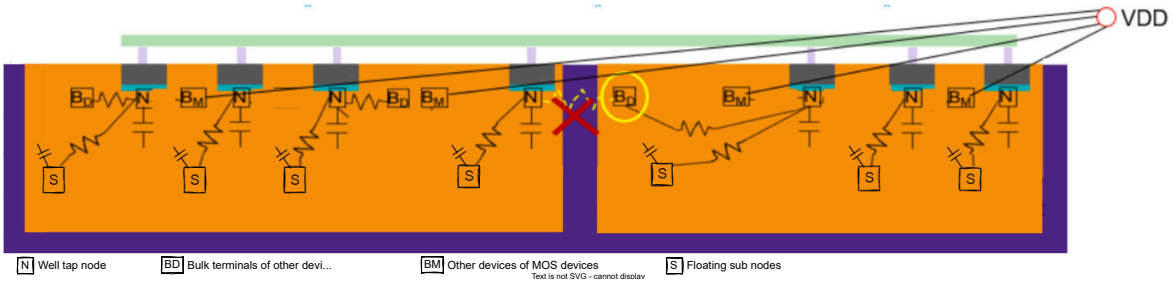
The CONLY BULK_THRU_WELLTAPS Option

The tool extracts the coupling capacitance to device bulk layers that serve as device terminal layers. For example, in an isolated SUBSTRATE layer, the StarRC tool identifies a group of bulk terminals and SUBSTRATE or well tap layers associated with the terminals. The SUBSTRATE or well tap layers are connected through metal interconnect, resulting in indiscriminate shorting between bulk and well tap layers and leading to shorts and loss of metal resistance. The following rules are considered to connect SUBSTRATE or well tap layers:

- In a well tap layer, no two SUBSTRATE or well tap layers can connect to each other.
- A bulk terminal is allowed to connect to *only one* terminal close to the SUBSTRATE or well tap layer.
- The capacitance to the SUBSTRATE layer should be distributed across various SUBSTRATE or well tap nodes
- If there are more SUBSTRATE or well tap layers than the bulk terminals, additional SUBSTRATE or well tap nodes do not connect to the bulk terminals by themselves.

Feedback

Figure 222 Illustration for TRANSLATE_RETAIN_BULK_LAYERS: CONLY BULK_THRU_WELLTAPS



The `CONLY BULK_THRU_WELLTAPS` command processes SUBSTRATE nodes in the following stages:

- During extraction: SUBSTRATE nodes are disconnected from each other and reconnected to the nearest well tap layer. The behavior is the same in the standard (xout) and GPD flows.
- Generating a netlist file: If the net was connected through SUBSTRATE layer only, the tool writes shorting resistors between different SUBSTRATE islands in the netlist file to prevent false opens caused by disconnection during extraction.

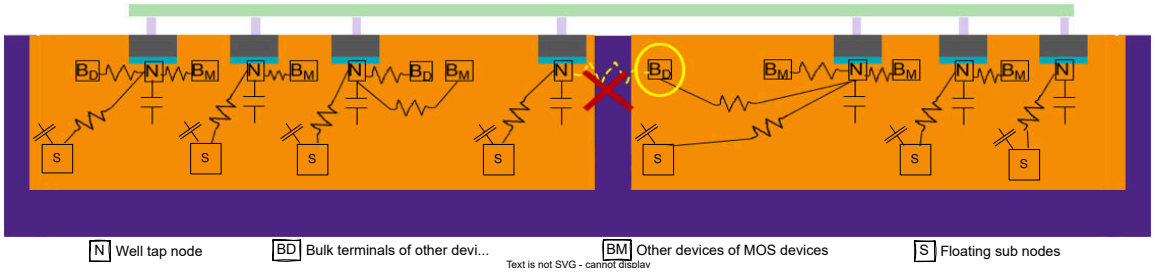
Note that when the SUBSTRATE or well tap layers are identified by the `KEEP_SUBCONT_MODELS` command, the tool connects the well tap layers together. This is similar to the behavior of the `CONLY` option. For information, see The CONLY Option.

When you use either the `CONLY BULK_THRU_WELLTAPS` or `CONLY BULK_THRU_WELLTAPS ALL_TO_CLOSEST_CONTACT` option, the StarRC tool preferably connects device bulk terminals to the power net port and connects other terminals on SUBSTRATE layers that are closest to substrate contacts nodes to do the following function:

- Retains all contacts for each SUBSTRATE layer.
- Avoids connecting the bulk terminals together, thereby avoiding shorting through the SUBSTRATE layer.
- Prevents opens to check connectivity through conductor layers and to add shorting resistor between SUBSTRATE nodes if connection is made through the SUBSTRATE layer in the design.

Feedback

Figure 223 Illustration for TRANSLATE_RETAIN_BULK_LAYERS: CONLY BULK_THRU_WELLTAPS ALL_TO_CLOSEST_CONTACT



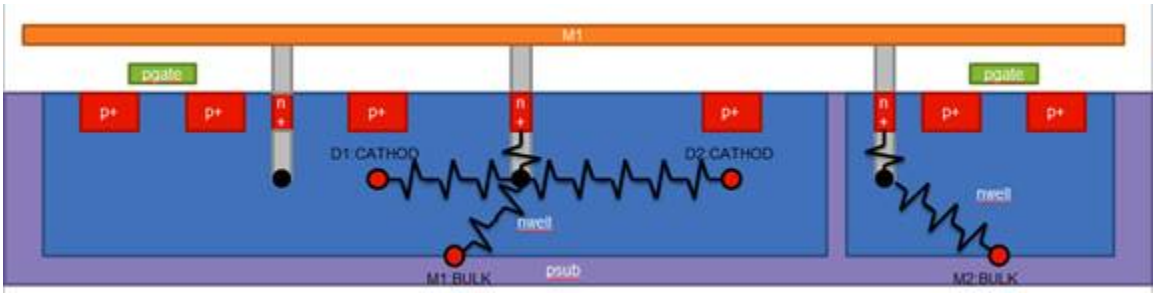
The CONLY ALL_TO_CLOSEST_CONTACT Option

The tool extracts the coupling capacitance to device bulk layers that serve as device terminal layers. In this mode, the bulk terminal is connected to the closest tap (contact between the substrate and a low-level metal layer). As a result, the bulk connection is reported as a *|I instance in the netlist, allowing back-annotation on bulk nodes by simulation tools.

If multiple floating taps connect to a bulk layer, only one tap is retained for every isolated connection.

Figure 224 illustrates the `CONLY ALL_TO_CLOSEST_CONTACT` option. The bulk layers are extracted for capacitance and connected to the closest contact.

Figure 224 Effect of the CONLY ALL_TO_CLOSEST_CONTACT Option



Interaction With Skip Cells

For a design that includes skip cells, special care must be taken to prevent undesirable shorting paths through the substrate.

Figure 225 illustrates the potential shorting path between two identical skip cells. The top-level port VSS connects to metal level M0_A, which connects to the cell instance port named GT. If the instance ports inside each cell connect to the substrate, the two ports connect through the substrate instead of through an upper-level metal layer, which is the expected behavior.

Feedback

To prevent such shorting paths, the INSTANCE_PORT_OPEN_CONDUCTANCE command does not insert shorting resistors between ports and the substrate inside skip cells. In addition, when the CONLY or CONLY ALL_TO_CLOSEST_CONTACT option is used for a design that includes skip cells, the StarRC tool retains only one substrate contact in a skip cell. In flat designs, the tool retains all substrate contacts.

Figure 225 Skip Cell Substrate Contact Handling

Examples

The following lines show an excerpt from a netlist generated using the TRANSLATE_RETAIN_BULK_LAYERS: CONLY command:

```
*|NET VDD 0.00463746PF
*|I (M2:SRC M2 SRC B 0 5 35.75)
Cg6 VDD:1 GND 1.55208e-15
...
*
*Instance Section
*
MM2 M2:DRN M2:GATE M2:SRC VDD nch ...
```

The following lines show an excerpt from a netlist generated using the TRANSLATE_RETAIN_BULK_LAYERS: CONLY ALL_TO_CLOSEST_CONTACT command:

```
*|NET VDD 0.00463746PF
*|I (M2:BULK M2 BULK B 0 5.5 35.75)
*|I (M2:SRC M2 SRC B 0 5 35.75)
Cg6 VDD:1 GND 1.55208e-15
...
*
*Instance Section
*
MM2 M2:DRN M2:GATE M2:SRC M2:BULK nch ...
```

See Also

- COUPLE_TO_GROUND
- INSTANCE_PORT_OPEN_CONDUCTANCE

- SHORT_PINS
- KEEP_SUBCONT_MODELS

Feedback

TRANSLATE_VIA_FILLS

Specifies whether to extract via fill objects.

Syntax

```
TRANSLATE_VIA_FILLS: YES | NO
```

Arguments

Argument	Description
YES	Translates via fills
NO (default)	Does not translate via fills

Description

Metal fill schemes sometimes include vias between the fill polygons on adjacent metal layers. These vias are known as fill vias or via fills. The resulting network of metal fill and via fill structures does not connect to any signal net.

Via fills typically do not have a significant impact on total capacitance.

The TRANSLATE_VIA_FILLS command specifies whether to translate via fills. This command is valid for all flows.

Examples

When you use the following command, the StarRC tool does not translate via fills.

```
TRANSLATE_VIA_FILLS: NO
```

Feedback

TRANSLATE_VIA_PINS

Specifies whether to translate incomplete via pins to real vias.

Syntax

```
TRANSLATE_VIA_PINS: YES | NO
```

Arguments

Argument	Description
YES (default)	Translates via pins to real vias
NO	Does not translate via pins to real vias. There is no impact on connectivity.

Description

An incomplete via is a via with a missing upper-layer net name or lower-layer net name. The TRANSLATE_VIA_PINS command specifies whether to translate via pins to real vias in the Milkyway flow.

Examples

When you use the following command, StarRC does not translate incomplete vias in the FRAM view.

```
TRANSLATE_VIA_PINS: NO
```

Feedback

TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO

Specifies the segmentation ratio of virtual vias in a trench contact process.

Syntax

```
TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO: ratio
```

Arguments

Argument	Description
ratio	Segmentation ratio represented as a floating-point number Units: none Default: 1

Description

Trench contacts can have tall covertical layers that are not connected by physical vias. To extract vertical resistance, virtual vias are inserted between the trench contact conductors. The StarRC tool segments these vias automatically to create a distributed resistance network.

The segmentation ratio is a floating-point number calculated as follows:

ratio = (via length) / (via width)

The height of the virtual via must be greater than or equal to 0.1 nm and less than or equal to 5 nm.

Errors

You can use the `TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO` command without the `MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER` command. However, if you use the `MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER` command alone, the StarRC tool issues an error message.

Examples

```
TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO: 2.5
```

See Also

- MAX_VIRTUAL_VIA_SEGMENTATION_NUMBER

Feedback

TSV_ CELLS

Specifies the TSV cells used for subcircuit replacement in the 3-D IC flow.

Syntax

```
TSV_ CELLS: tsvcell_1 tsvcell_2 tsvcell_3 ...
```

Arguments

Argument	Description
`tsvcell_1` ...	TSV cells used for subcircuit replacement in the 3-D IC flow

Description

The `TSV_CELLS` command specifies the TSV cells used for subcircuit replacement in the 3-D IC flow.

Examples

The following syntax specifies the TSV cells used for subcircuit replacement.

```
TSV_ CELLS: tsvcell_1 tsvcell_2
```

See Also

- TSV
- 3D_IC_FILTER_DEVICE
- 3D_IC_FLOATING_SUBSTRATE
- 3D_IC_SUBCKT_FILE
- 3D_IC_TSV_COUPLING_EXTRACTION
- Through-Silicon Vias

Feedback

USER_DEFINED_DIFFUSION_RES

Enables user-defined diffusion resistance calculation. Valid only for transistor-level flows.

Syntax

```
USER_DEFINED_DIFFUSION_RES: YES | NO
```

Arguments

Argument	Description
YES (default)	Enables user-defined diffusion resistance calculation
NO	Disables all use of user-defined diffusion resistance

Description

The USER_DEFINED_DIFFUSION_RES command enables the calculation of diffusion resistance by a method that is more accurate for advanced process nodes than the standard mesh resistance calculation.

Note:

This implementation of user-defined diffusion resistance is not compatible with the feature of the same name available in earlier StarRC versions. To use this feature, you must use an nxtgrd file created in StarRC version L-2016.06 or later.

To use this analysis, perform the following steps:

1. Set the USER_DEFINED_DIFFUSION_RES command to YES in the StarRC command file. (This step is optional because the command defaults to YES.)
2. Include the USER_DEFINED_DIFFUSION_RESISTANCE command within the CONDUCTOR block of the gate conductor layer in the ITF file.
3. Set the LAYER_TYPE keyword to GATE within the CONDUCTOR block of the gate conductor layer in the ITF file.
4. Specify a model name by using the diffusion_res_model keyword in the conducting_layers command in the mapping file

Feedback

See Also

- USER_DEFINED_DIFFUSION_RESISTANCE
- conducting_layers
- User-Defined Diffusion Resistance

Feedback

VERTICAL_GATE_RESISTANCE

Specifies whether to model vertical gate resistance.

Syntax

```
VERTICAL_GATE_RESISTANCE: YES | NO
```

Arguments

Argument	Description
`YES` (default)	Models vertical gate resistance
`NO`	Disables vertical gate resistance analysis

Description

By default, the StarRC tool models the vertical resistance of transistor gates if the following conditions are met in the nxtgrd file:

- The gate layer is a conductor layer that includes a `LAYER_TYPE = GATE` specification.
- The gate conductor definition includes a `VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH` table.

For backward compatibility, disable modeling of these vertical resistances by specifying `VERTICAL_GATE_RESISTANCE: NO` in the StarRC command file.

See Also

- LAYER_TYPE
- VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH

Feedback

VIA_ARRAY_COUNT_FILE

Specifies a file that defines the maximum number of vias to merge for specific layers. Valid for transistor-level flows only.

Syntax

```
VIA_ARRAY_COUNT_FILE: via_count_file
```

Arguments

Argument	Description
via_count_file	A file that contains layer names and the maximum number of vias in either the horizontal or vertical direction to merge for those layers Default: none

Description

The VIA_ARRAY_COUNT_FILE command specifies the maximum number of vias to merge on a specific database layer in transistor-level flows. The via count in the file applies to both the horizontal and vertical directions of the via array.

For example, if the via count is 3, the tool locates a 3x3 array of vias, merges that array into a single via, then examines additional vias in both directions and continues to merge 3x3 arrays into single vias.

Each line in the file must use the following syntax:

```
database_layer_name  number_of_vias
```

The spacing between vias to be merged must meet the following conditions:

- The spacing between vias must be less than or equal to twice the square root of the via area.
- The horizontal spacing between vias must be the same; the vertical spacing between vias must be the same. However, the horizontal spacing can be different from the vertical spacing.

You can also specify this information directly by using the VIA_ARRAY_COUNT_LAYER command. If the same layer is specified in both a VIA_ARRAY_COUNT_LAYER command and a file specified by a VIA_ARRAY_COUNT_FILE command, the VIA_ARRAY_COUNT_LAYER command takes precedence.

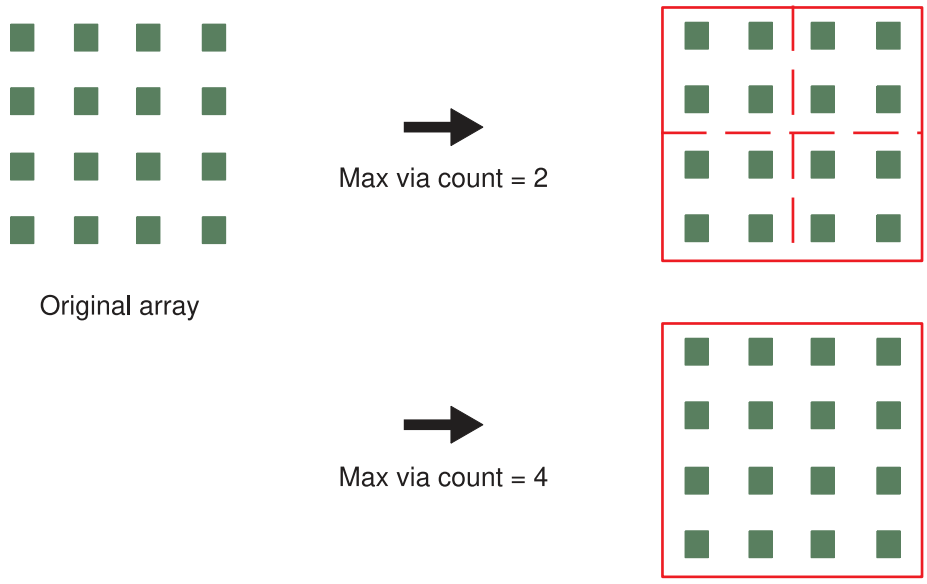
Via merging for layers specified in a via array count file occurs even if the MERGE_VIAS_IN_ARRAY command is set to NO.

Feedback

If a layer is not specified in either a via array count file or a VIA_ARRAY_COUNT_LAYER command, via merging for that layer occurs according to the rules determined by the MERGE_VIAS_IN_ARRAY command and the via_layers statement in the mapping file.

Figure 226 shows the effect of the maximum via count on via merging.

Figure 226 Via Merging



Errors

If a maximum via count is specified and the via_layers statement for the same layer in the mapping file contains the MAX_VIA_ARRAY_SPACING or MAX_VIA_ARRAY_LENGTH options, the StarRC tool issues a warning message and ignores the MAX_VIA_ARRAY_SPACING and MAX_VIA_ARRAY_LENGTH options.

If the StarRC command file contains a VIA_COVERAGE_OPTION_FILE or VIA_COVERAGE command, the VIA_ARRAY_COUNT_FILE command is ignored.

See Also

- VIA_ARRAY_COUNT_LAYER
- MERGE_VIAS_IN_ARRAY
- via_layers

Feedback

VIA_ARRAY_COUNT_LAYER

Specifies the maximum number of vias to merge for specific layers. Valid for transistor-level flows only.

Syntax

```
VIA_ARRAY_COUNT_LAYER: layer1 n1 layer2 n2 ...
```

Arguments

Argument	Description
layer1, layer2, ...	A database layer name
n1, n2, ...	The maximum number of vias to merge for the corresponding layer; an integer greater than 1

Description

The VIA_ARRAY_COUNT_LAYER command specifies the maximum number of vias to merge on a specific database layer in transistor-level flows. The specified via count applies to both the horizontal and vertical directions of the via array.

For example, if the via count is 3, the tool locates a 3x3 array of vias, merges that array into a single via, then examines additional vias in both directions and continues to merge 3x3 arrays into single vias.

The spacing between vias to be merged must meet the following conditions:

- The spacing between vias must be less than or equal to twice the square root of the via area.
- The horizontal spacing between vias must be the same; the vertical spacing between vias must be the same. However, the horizontal spacing can be different from the vertical spacing.

You can also provide this information in a separate file specified with the VIA_ARRAY_COUNT_FILE command. If the same layer is specified in both a file and a VIA_ARRAY_COUNT_LAYER command,, the VIA_ARRAY_COUNT_LAYER command takes precedence.

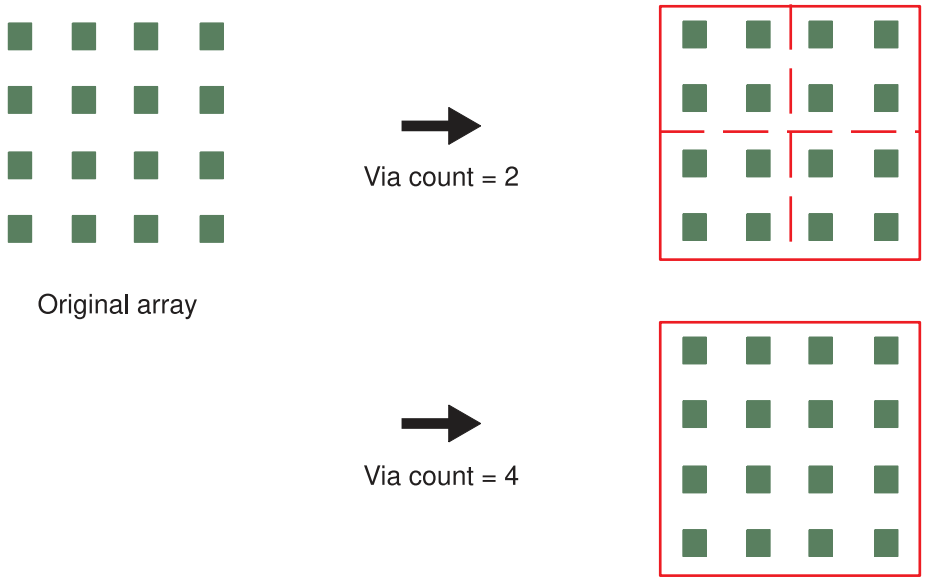
Via merging for layers specified in this command occurs even if the MERGE_VIAS_IN_ARRAY command is set to NO.

If a layer is not specified in either a via array count file or a VIA_ARRAY_COUNT_LAYER command, via merging for that layer occurs according to the rules determined by the

MERGE_VIAS_IN_ARRAY command and the via_layers statement for the via layer in the mapping file.

Figure 227 shows the effect of the maximum via count on via merging.

Figure 227 Via Merging



Errors

If a maximum via count is specified and the via_layers statement for the same layer in the mapping file contains the MAX_VIA_ARRAY_SPACING or MAX_VIA_ARRAY_LENGTH options, the StarRC tool issues a warning message and ignores the MAX_VIA_ARRAY_SPACING and MAX_VIA_ARRAY_LENGTH options in the mapping file.

If the StarRC command file contains a VIA_COVERAGE_OPTION_FILE or VIA_COVERAGE command, the VIA_ARRAY_COUNT_LAYER command is ignored.

Examples

The following command specifies a maximum via merge count of 3 for layer via4 and a maximum via merge count of 2 for layer via5:

```
VIA_ARRAY_COUNT_Layer: via4 3 via5 2
```

Feedback

See Also

- VIA_ARRAY_COUNT_FILE
- MERGE_VIAS_IN_ARRAY
- via_layers

Feedback

VIA_COVERAGE

Specifies via coverage analysis for square vias.

Syntax

```
VIA_COVERAGE: via1 Lf Lq [Ls] Cf Cq [Cs]
```

Arguments

Argument	Description
Lf	Maximum number for a landing full measurement Units: nanometers
Lq	Maximum number for a landing quarter measurement Units: nanometers
Ls	Maximum number for a landing semi measurement Units: nanometers
Cf	Maximum number for a coverage full measurement Units: nanometers
Cq	Maximum number for a coverage quarter measurement Units: nanometers
Cs	Maximum number for a coverage semi measurement Units: nanometers

Description

You can enable via coverage analysis and define the coverage parameters in either of the following ways:

- Specify the `VIA_COVERAGE` command in the ITF file (nxtgrd file).
- Use one or both of the `VIA_COVERAGE` and `VIA_COVERAGE_OPTION_FILE` commands in the StarRC command file.

If the ITF file contains a `VIA_COVERAGE` command and the StarRC command file contains either the `VIA_COVERAGE` or `VIA_COVERAGE_OPTION_FILE` command, the tool issues an error message.

The `VIA_COVERAGE` command checks square vias; the `VIA_COVERAGE_OPTION_FILE` command checks rectangular vias. Each number corresponds to a coverage or landing measurement.

Feedback

Note:

The NETLIST_TAIL_COMMENTS:YES command is required; the NETLIST_FORMAT:SPEF command is not required. The VIA_COVERAGE command does not require a text file.

Each VIA_COVERAGE command must have all six entries to include the semicoverage capability. You can specify four numbers to get results without the semicoverage capability. Both are reported in the netlist under the heading “VIA_COVERAGE_CODES.”

The coverage and landing units are nanometers and represent as-drawn dimensions, before the application of any MAGNIFICATION_FACTOR command.

The full coverage and landing values must be greater than the semicoverage and landing values. All vias specified for this feature must also be defined in the ITF file.

Examples

This example specifies the measurement of via coverage including semicoverage values.

```
NETLIST_FORMAT: SPF
NETLIST_TAIL_COMMENTS: YES
VIA_COVERAGE: via1 100 80 100 80
VIA_COVERAGE: via2 100 80 100 80
```

See Also

- MERGE_VIAS_IN_ARRAY
- NETLIST_FORMAT
- NETLIST_TAIL_COMMENTS
- VIA_COVERAGE_OPTION_FILE
- Via Coverage

Feedback

VIA_COVERAGE_OPTION_FILE

Specifies a file that contains via checking rules for rectangular vias.

Syntax

```
VIA_COVERAGE_OPTION_FILE: file_name
```

Arguments

Argument	Description
file_name	The via coverage option file

Description

You can enable via coverage analysis and define the coverage parameters in either of the following ways:

- Specify the VIA_COVERAGE command in the ITF file (nxtgrd file).
- Use one or both of the VIA_COVERAGE and VIA_COVERAGE_OPTION_FILE commands in the StarRC command file.

If the ITF file contains a VIA_COVERAGE command and the StarRC command file contains either the VIA_COVERAGE or VIA_COVERAGE_OPTION_FILE command, the tool issues an error message.

The VIA_COVERAGE_OPTION_FILE command provides the name of a file that contains rules for checking rectangular vias. The command can be used in the StarRC command file or optionally within a corners file for a simultaneous multicorner flow.

The via coverage report in the netlist contains a table that shows the detail of the via coverage results. The via coverage results are shown as full coverage, quarter coverage, semicoverage, and partial coverage.

A via satisfies the area check of a drawn box if the box area in the figure is filled with metal polygons. This applies to both coverage and landing. For coverage, the metal layer refers to the metal layer above the via layer (for example, metal2 for via12) and for the landing it refers to the metal layer below the via layer (for example metal1 for via 12).

The default dbunit for the values in the via coverage option file is 1000 (in other words, the units are nanometers).

Feedback

The output netlist contains the following via classifications:

- `FULL` means all sides of the via are covered because all enclosures are greater than the F parameter (as shown in Figure 228).
- `QUARTER` means one enclosure must be greater than or equal to Q1 and must have both adjacent sides enclosed by greater than Q2.
- `SEMI` means one enclosure must be greater than or equal to S1 and both adjacent sides must be enclosed by more than S2.
- `PARTIAL` means no enclosures meet the full, quarter, or semicoverage requirements.

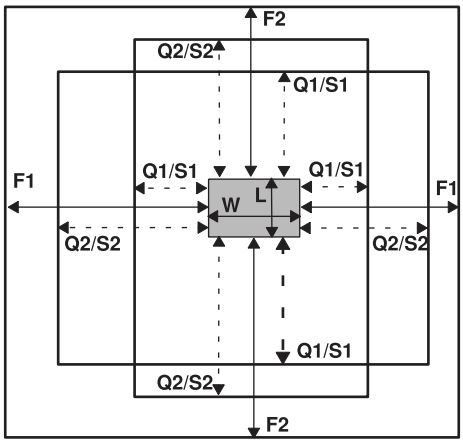
Specify three numbers for coverage and three numbers for landing when you are not checking semicoverage and landing. Specify five numbers for coverage and five numbers for landing when you are checking semicoverage and landing.

Figure 228 Via Coverage

W = width of the VIA in the horizontal direction

L = length of the VIA in the vertical direction

F1, F2, Q2, Q1, S2, S1 are the parameters in the `VIA_COVERAGE_OPTION_FILE` command



The `VIA_COVERAGE_OPTION_FILE` command rules are as follows:

- Non-Manhattan shapes are not supported.
- For via arrays, all inside vias (those not on the perimeter) are considered fully covered.
- The horizontal direction is equal to the direction of the x-axis of coordinates.
- The vertical direction is equal to the direction of the y-axis of coordinates.

- Via coverage parameters must meet the following conditions, which apply to both coverage and landing parameters:
  - F must be greater than or equal to Q2.
  - F must be greater than or equal to S2.
  - Q2 must be greater than S2.
  - Q2 must be greater than Q1.
  - S2 must be greater than or equal to S1.

The following table lists rules for the parameters.

Keyword	Description
`Xrange`	Width of the via contact
`Xmin`	Minimum width value of the via contact
`Xmax`	Maximum width value of the via contact
`Yrange`	Length of the via contact
`Ymin`	Minimum length value of the via contact
`Ymax`	Maximum length value of the via contact
`FL1`	Full coverage y value for via landing
`FL2`	Full coverage x value for via landing
`FC1`	Full coverage y value for via cover
`FC2`	Full coverage x value for via cover
`QL1`	Quarter coverage value for landing (small enclosure value)
`QC1`	Quarter coverage value for via cover (small enclosure value)
`QL2`	Quarter coverage value for via landing (large enclosure value)
`QC2`	Quarter coverage value for via cover (large enclosure value)
`SL1`	Semicoverage value for via landing (small enclosure value)
`SC1`	Semicoverage value for via cover (small enclosure value)
`SL2`	Semicoverage value for via landing (large enclosure value)
`SC2`	Semicoverage value for via cover (large enclosure value)

When the VIA_COVERAGE_OPTION_FILE command is used in a corners file for a simultaneous multicorner extraction, the following rules apply:

- If two corners have the same nxtgrd file, they must use the same via coverage option file. However, different nxtgrd files can use the same via coverage option file.
- The only variations allowed within the via coverage option files are RPV variations.
- If any corner has a via coverage option file, all corners must have such a file.
- If via coverage option files are defined in the corners file, a VIA_COVERAGE_OPTION_FILE command in the global command file is ignored.
- If no via coverage option files are defined in the corners file, a VIA_COVERAGE_OPTION_FILE command in the global command file is applied.
- Via resistance is calculated based on the RPV value for each process corner. If temperature deration is defined for the associated via layer, the deration is applied to each corner-dependent RPV value.

Examples

In the following example files, Xrange and Yrange represent the as-drawn length in X and Y dimensions of the via (before any scale factor has been applied). The ranges must not be overlapping, and each via size should be mappable to exactly one of the data sets in the list. The range specification for a via landing and via coverage is the same. These requirements are checked in the tool. No interpolation or extrapolation is needed. If a via is found that cannot be mapped to one of the specified ranges, the StarRC tool issues a warning and gives the via an “invalid” via coverage code of 0. There is no limit to the number of data set ranges.

Text File Syntax Single Parameter

```
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing = FL,QL1,QL2,[SL1,SL2];
Coverage = FC,QC1,QC2,[SC1,SC2]}
(Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;
Landing = FL,QL1,QL2,[SL1,SL2];Coverage = FC,QC1,QC2,[SC1,SC2]}
```

Text File Syntax With Two Parameters

```
Without semicoverage extraction
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;Landing = FL1,FL2,QL1,QL2;
Coverage = FC1,FC12,QC1,QC2}

With semicoverage extraction
via_layer_name
{Xrange=Xmin,Xmax;Yrange=Ymin,Ymax;
Landing = FL1,FL2,QL1,QL2,[SL1,SL2];Coverage = FC1,FC2,QC1,QC2,[SC1,SC2]}
```

Feedback

File Format Example 1 - With Semicoverage

```
VIA1 { Xrange= 100,100; Yrange = 100,100;
Landing = 100,80,10,40,10;Coverage = 100,80,10,40,10}
```

File Format Example 1 - Without Semicoverage

```
VIA1 { Xrange= 100,100; Yrange = 100,100;
Landing = 100,80,10;Coverage = 100,80,10}
```

See Also

- NETLIST_FORMAT
- NETLIST_TAIL_COMMENTS
- REDUCTION
- VIA_COVERAGE
- SIMULTANEOUS_MULTI_CORNER
- Via Coverage

Feedback

VIA_SMIN

Specifies the minimum spacing between vias. Valid only in transistor-level electromigration flows.

Syntax

```
VIA_SMIN: layer1 smin1 layer2 smin2 ...
```

Arguments

Argument	Description
layer1, layer2, ...	A database layer name for a via layer
smin1, smin2, ...	The minimum spacing between vias Units: microns

Description

The VIA_SMIN value is used only in transistor-level electromigration flows. The tool uses the VIA_SMIN value along with the VIA_WMIN value to calculate the maximum allowable spacing between vias for via merging.

You can also provide these values by using the SMIN and WMIN options for the via layer in the ITF file. Values in the command file take precedence over values in the ITF file.

If the WMIN and SMIN statements are missing from the via layer definition in the ITF file and VIA_SMIN and VIA_WMIN are missing from the StarRC command file, default via merging occurs. If only one set of values is provided, the tool uses these values in the new flow.

In the electromigration flow, the tool always performs via merging and ignores the MERGE_VIAS_IN_ARRAY command.

See Also

- VIA_WMIN
- SMIN
- WMIN
- Via Merging

Feedback

VIA_SUM_LIMIT

Limits the number of via violations to include in summary reports.

Syntax

```
VIA_SUM_LIMIT: max_count
```

Arguments

Argument	Description
max_count	Maximum number of via violations to report Default: 1000

Description

In cases where the StarRC tool identifies a large number of via violations, you might want to limit the amount of detail included in summary reports and the number of times that similar messages are issued.

The `VIA_SUM_LIMIT` command specifies the maximum number of unique via violations for which to report detailed information such as layer names and bounding boxes in the *vias.sum* file. If the limit is exceeded, the StarRC tool writes a warning message in the file and does not report the additional violations.

See Also

- Via Violations Reports

Feedback

VIA_WMIN

Specifies the minimum via width. Valid only in transistor-level electromigration flows.

Syntax

```
VIA_WMIN: layer1 wmin1 layer2 wmin2 ...
```

Arguments

Argument	Description
layer1, layer2, ...	A database layer name for a via layer
wmin1, wmin2, ...	The minimum via width Units: microns

Description

The VIA_SMIN value is used only in transistor-level electromigration flows. The tool uses the VIA_SMIN value along with the VIA_WMIN value to calculate the maximum allowable spacing between vias for via merging.

You can also provide these values by using the SMIN and WMIN options for the via layer in the ITF file. Values in the command file take precedence over values in the ITF file.

If the WMIN and SMIN statements are missing from the via layer definition in the ITF file and VIA_SMIN and VIA_WMIN are missing from the StarRC command file, default via merging occurs. If only one set of values is provided, the tool uses these values in the new flow.

In the electromigration flow, the tool always performs via merging and ignores the MERGE_VIAS_IN_ARRAY command.

See Also

- VIA_SMIN
- SMIN
- WMIN
- Via Merging

Feedback

VIOLATION_REPORT_SPEF_ESCAPING

Controls whether the escape (\) character should be present in the net names of the shorts and opens summary files that are in the star directory.

Syntax

```
VIOLATION_REPORT_SPEF_ESCAPING: YES | NO | FOLLOW_NETLIST_FORMAT
```

Arguments

Argument	Description
`FOLLOW_NETLIST_FORMAT` (default)	Allows the escape character in the net names of the shorts and opens summary files in the STAR directory only if the netlist format is set to `SPEF` with the `NETLIST_FORMAT:SPEF` command in the StarRC command file
`YES`	Allows the escape character in the net names of the shorts and opens summary files
`NO`	Does not allow the escape character in the net names of the shorts and opens summary files

Description

The command controls whether the escape character should be present in the net names of the shorts and opens summary files. The presence of the escape character in the net names is based on the directory where the shorts and opens summary files are stored:

- Star directory: The escape character is in the net names of the shorts and opens summary files only if the netlist format is set to `SPEF` with the `NETLIST_FORMAT` command. You can choose to change the net names by setting the behavior of the escape character with the `VIOLATION_REPORT_SPEF_ESCAPING` command.
- GPD directory: The escape character is not in the net names of the short and opens summary files.

Feedback

VIRTUAL_METAL_FILL_EXCLUDED_CELLS

Specifies a list of cells from which to exclude virtual metal fill placement.

Syntax

```
VIRTUAL_METAL_FILL_EXCLUDED_CELLS: cell1 cell2 ...
```

Arguments

Argument	Description
cell1 cell2 ...	Cells to exclude from virtual metal fill insertion Default: none

Description

Before the insertion of real metal fill, you can insert virtual metal fill polygons for better estimation of parasitics and timing early in the design process. Virtual metal fill emulates track-fill style metal fill using large aspect ratio polygons.

Caution:

Virtual metal fill is not intended for use in a signoff flow because it does not represent the actual fill in a design.

To exclude cells from virtual metal fill placement, list the cells in the `VIRTUAL_METAL_FILL_EXCLUDED_CELLS` command. If you specify a cell for exclusion, you can optionally include the `fill_blockage_excluded_cells` parameter in the parameter file to specify the distance of virtual metal fill polygons from the excluded cells.

See Also

- VIRTUAL_METAL_FILL_NDR_NETS
- VIRTUAL_METAL_FILL_POLYGON_HANDLING
- VIRTUAL_METAL_FILL_PARAMETERIZE
- VIRTUAL_METAL_FILL_PARAMETER_FILE
- Virtual Metal Fill

Feedback

VIRTUAL_METAL_FILL_NDR_NETS

Specifies a list of nondefault rule nets for special treatment with virtual metal fill.

Syntax

```
VIRTUAL_METAL_FILL_NDR_NETS: net1 net2 ...
```

Arguments

Argument	Description
net1 net2 ...	Nets to treat as NDR nets in virtual metal fill creation Default: none

Description

The VIRTUAL_METAL_FILL_NDR_NETS command specifies a list of nondefault rule (NDR) nets for special handling in the virtual metal fill flow.

Caution:

Virtual metal fill is not intended for use in a signoff flow because it does not represent the actual fill in a design.

Virtual metal fill polygons are defined by the VIRTUAL_METAL_FILL_PARAMETER_FILE and VIRTUAL_METAL_FILL_POLYGON_HANDLING commands.

The min_fill_ndr_w_spacing and fill_ndr_l_spacing parameters in the virtual metal fill parameter file are used only for nets specified with the VIRTUAL_METAL_FILL_NDR_NETS command.

If spacing rules for NDR nets are defined in the design database and those spacings are larger than the min_fill_ndr_w_spacing and fill_ndr_l_spacing parameters, the rules in the design database take priority.

An NDM format design created with the Fusion Compiler or IC Compiler II tool might also contain virtual shielding rules. If the INDESIGN_VIRTUAL_SHIELDING command is set to YES in the StarRC command file, the StarRC tool honors the virtual shielding rules when determining how to place virtual metal fill polygons.

Some nets might have multiple sets of NDR rules defined. For example, the VIRTUAL_METAL_FILL_NDR_NETS command might specify nets that already have NDR rules in the design database. In this case, the tool uses the largest spacing from all rules when inserting metal fill polygons near an NDR net.

Feedback

See Also

- INDESIGN_VIRTUAL_SHIELDING
- VIRTUAL_METAL_FILL_POLYGON_HANDLING
- VIRTUAL_METAL_FILL_PARAMETER_FILE
- VIRTUAL_METAL_FILL_PARAMETERIZE
- VIRTUAL_METAL_FILL_EXCLUDED_CELLS
- Virtual Metal Fill

Feedback

VIRTUAL_METAL_FILL_OPTIONS_FILE

Specifies a file with complex parameters and rules to drive virtual metal fill extraction and achieve better correlation with real metal fill characteristics.

Syntax

```
VIRTUAL_METAL_FILL_OPTIONS_FILE: file_name
```

Arguments

Argument	Description
file_name	Virtual metal fill options file with design-specific parameters

Description

Before inserting real metal fill, you can insert virtual metal fill polygons for better estimation of parasitics and timing early in the design process.

Caution:

Virtual metal fill is not intended for use in a signoff flow because it does not represent the actual fill in a design.

The virtual metal fill options file is written in JavaScript Object Notation (JSON) format to improve readability and usability, with respect to the virtual metal fill parameter file format. The content of the virtual metal fill options file is similar to the virtual metal fill parameter file. However, you can specify `VIRTUAL_METAL_FILL_OPTIONS_FILE` command, for example,

- Multiple virtual metal fill placements on the same design, the same layer, and the same region
- Asymmetric placement rules
- Each layer handled as needed
- Region-based skip or placement virtual metal fill rules

Note:

The `VIRTUAL_METAL_FILL_OPTIONS_FILE` command settings overrides the `VIRTUAL_METAL_FILL_PARAMETER_FILE` command settings. You must also use the `VIRTUAL_METAL_FILL_POLYGON_HANDLING` command when you provide a virtual metal fill options file.

Feedback

When you use the VIRTUAL_METAL_FILL_OPTIONS_FILE command,

- Specify each layer either with the grounded or floating setting to handle the virtual metal fill polygons. This setting overrides the VIRTUAL_METAL_FILL_POLYGON_HANDLING command settings.
- Specify bounding boxes, based on which the tool excludes or limits the creation of virtual metal fills for all layers or for each layer.
- Specify on which layers and for which regions the tool should perform virtual metal fill. This setting overrides the VIRTUAL_METAL_FILL_PARAMETER_FILE command settings.
- Place virtual metal fill polygons in multiple passes to achieve better QoR

The StarRC tool accepts the following file types for the virtual metal fill options file:

- A metal fill parameter file generated by Fusion Compiler or IC Compiler II in the fill run directory track_fill_params.rh, which is recognized by the presence of the layer mapping definition. You can specify multiple metal fill runset files in the argument list of the VIRTUAL_METAL_FILL_OPTIONS_FILE command. In this case, the first file in the list must contain the layer mapping information.
- An ASCII file is generated by the Fusion Compiler or IC Compiler II tool with the set_extraction_options -virtual_metalfill_starrc_options_file command. You can specify only one of these files in the argument list.
- A file is generated by the StarRC tool when you set the VIRTUAL_METAL_FILL_PARAMETERIZE command to YES. Use this command to analyze real metal fill and write an options file based on its layout. You can analyze a metal fill design strategy one time. Then, use the generated virtual metal fill options file for subsequent extraction runs for the same technology.
- A manually-created ASCII file must have the first line to specify the version of the virtual metal fill options file to create virtual metal fill. See Table 94 for VMF_options_file_version.

The StarRC tool checks the validity of all the sections and issues warning or error messages as needed. Table 94 describes each section specified in the virtual metal file options file.

```
// Specify the version change of the options file
"VMF_options_file_version" : "1.0.0"

// Define properties at global level
"global": {
   // Do not place virtual metal fill on regions between [15 30] [20 40]
   "region": {
      "name" : "global_region_1"
      "type" : "skip"
      "bbox" : [15, 20, 30, 40]
```

Feedback

```
      }
   }

"layer": {
   "name":"M1"
   // Place regular virtual metal fill only between [10 15] [20 30] (if
 not present place on all design,
   excluded skip regions)
   "region": {
      "name" : "region_m1_a"
      "bbox" : [10, 20, 15, 30]
      }

   // Redefine placement options (if not present, use the global ones)
   "placement" : {
      "name":"placementA",
      "region":"region_m1_a",
      "direction":"H",
      "fill_width":0.04,
      "min_fill_length":0.10,
      "max_fill_length":1.00,
      "fill_route_w_spacing":0.05,
      "fill_route_l_spacing":0.06,
      "fill_fill_w_spacing":0.05,
      "fill_fill_l_spacing":0.06,
      "fill_blockage_w_spacing":0.07,
      "fill_blockage_l_spacing":0.08,
      "fill_ndr_w_spacing":0.06,
      "fill_ndr_l_spacing":0.03,
      "fill_blockage_excluded_cells":0.02,
      "asymmetric":"no", // default
      "polygon_handling_mode":"grounded",
      "algorithm": "adjustable_fill_fill"
      "enable":"yes",
      }
   }

"layer": {
   "name":"M2*"
   // Place regular virtual metal fill only between [10 15] [20 30] (if
 not present place on all design,
   excluded skip regions)
   "region": {
      "name" : "region_m2_a"
      "bbox" : [10, 20, 15, 35]
      }
   "region": {
      "name" : "region_m2_b"
      "bbox" : [40, 60, 95, 135]
      }

   // Redefine placement options (if not present, use the global ones)
   "placement" : {
```

```
        "name":"placementC",
        "region":"region_m2_a",
        "direction":"H",
        "fill_width":0.04,
        "min_fill_length":0.10,
        "max_fill_length":1.00,
        "fill_route_w_spacing":0.05,
        "fill_route_l_spacing":0.06,
        "polygon_handling_mode":"floating"
        }
    "placement" : {
        "name":"placementD",
        "region":"region_m2_b",
        "direction":"V",
        "fill_width":0.03,
        "min_fill_length":0.50,
        "max_fill_length":1.50,
        "fill_route_w_spacing":0.05,
        "fill_route_l_spacing":0.06,
        "polygon_handling_mode":"grounded"
        }
    }
```

Table 94 Section in Virtual Metal Fill Options File

Section	Description
`VMF_options_file_version`	The first line within the quotation marks to specify the version number of the options file. This line must be present in the virtual metal fill options file, so the tool recognizes the file format and interprets it appropriately.
“`global`”	Specify settings that apply to all design regions of all the layers only in this section.
“`layer`”	Defines a layer section for each layer with its region and placement sections. If you specify the layer section multiple times for the same layer, the settings of the last layer section overrides the settings specified in all the previous layer sections.
“`region`”	Defines a region to place or exclude virtual metal fill polygons in the global or layer section The region section has the following fields • “`name`”: Specify an unique name for the region. • (Optional) “`type`”: Specify the region type, `skip` or `place` (the default). • “`bbox`”: Specify coordinates of bounding box in microns.

Feedback

Table 94 Section in Virtual Metal Fill Options File (Continued)

Section	Description
	The tool places or excludes virtual metal fill polygons based on the following criteria: • The bounding box setting in a layer section overrides the bounding box settings in the global section. • If required fields are only specified, the tool excludes or performs placement of virtual metal fill polygons only in a design region. However, if other options with the required fields are specified in the same layer section, the settings are applied only to a specific design region.
"placement"	Defines a placement region to place virtual metal fill polygons. If a placement region is not specified, the entire design region is considered to place virtual metal fill polygons. To define parameters in the placement section, see the parameters listed in Table 95 in VIRTUAL_METAL_FILL_PARAMETER_FILE. **Note:** You can use all the parameters from the virtual metal fill parameter file in the placement section, except for the layer_name parameter. You can also specify the following statements: • "asymmetric": Places virtual metal fill polygons in the horizontal direction on a region by default. If you have specified multiple placements on the specified layer and on the specified region or on an overlapping region, the tool places virtual metal fill polygons in the sequential order as defined in the placement section. • "polygon_handling_mode": Specifies the treatment of virtual metal fill polygons either as grounded or floating (the default) on the specified layer. If specified, this setting overrides the VIRTUAL_METAL_FILL_PARAMETER_FILE and VIRTUAL_METAL_FILL_POLYGON_HANDLING commands settings. • g"algorithm": Specifies a placement algorithm to evaluate the impact of metal fills in the worst case scenario, for example, metal fill polygons closer to routing signal nets might cause significant impact on parasitics. For the list of parameters to specify with the algorithm statement, see Specifying Placement Algorithm.

Note:

To enable or disable a placement setting, use the "enable": "yes" key in the layer, region, or placement section.

Feedback

Handling Overlapping Regions

If you have defined overlapping regions on the same layer, the tool handles the regions as follows:

- If multiple place or skip regions are defined in an overlapping region, the skip region takes precedence over any place region.
- If only multiple place regions are defined, the placement order defined in the virtual metal fill options file determines the precedence to place virtual metal fill polygons.

Placing Virtual Metal Fill Polygons Asymmetrically in a Filling Region

The IC Validator tool by default creates configurations for asymmetric metal fill polygons to place metal fills in a placement region. To control placing of virtual metal fill polygons asymmetrically, use the symmetry option in the IC Validator tool to specify the fill pattern as center in a specified fill region. The default is `false`. Then, the IC validator tool

1. Does not center the fill pattern in the region
2. Places the stripe starting from the bottom-left corner of the region

If set to `true`, the IC Validator tool

1. Centers the fill pattern in a region
2. Places the first stripe with its center-line on the center of the bounding box
3. Places subsequent stripes at equal distances from the center stripe

The virtual metal fill options file can be edited to make the following changes that are done by the IC Validator tool:

- Specify to place virtual metal fill polygons asymmetrically
- Specify if the StarRC tool should center the fill pattern in the region for a specified layer

To avoid recentering of the fill pattern, use the `"asymmetric": "yes"` key in the placement section as shown in the following example. The StarRC tool places virtual metal fills starting from the bottom-left corner of the region according to the default asymmetric configuration of the IC Validator tool.

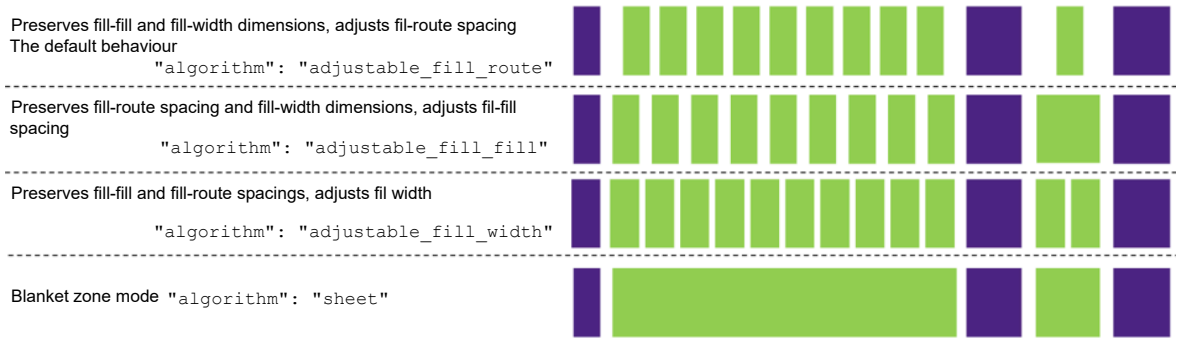
```
"placement" : {
    "name":"placementA",
    "region":"my_region2",
    "direction":"H", // or V
    "fill_width":0.04,
    "min_fill_length":0.10,
    "max_fill_length":1.00,
    "fill_route_w_spacing":0.05,
    "fill_route_l_spacing":0.06,
    "fill_fill_w_spacing":0.05,
```

```
    "fill_fill_l_spacing":0.06,
    "fill_blockage_w_spacing":0.07,
    "fill_blockage_l_spacing":0.08,
    "fill_ndr_w_spacing":0.06,
    "fill_ndr_l_spacing":0.03,
    "fill_blockage_excluded_cells":0.02,
    "asymmetric":"yes", // default is yes
    "polygon_handling_mode":"grounded"
}
```

Specifying Placement Algorithm

The algorithm for placing of virtual metal fill polygons in the design can be specified with the `algorithm` statement. For example, you can choose to preserve the fill-routing distance to evaluate the impact of parasitics in the worst conditions, where fill polygons are placed closer to routing signals. Note that in any case the tool does not place the virtual metal fill polygon if the space is smaller than the fill width.

Figure 229 Placement algorithm parameters



Use the following parameters to specify with the `algorithm` statement. Figure 229 shows the placements of virtual metal fill polygons with the different algorithms.

- `adjustable_fill_route` (the default)

  Places virtual metal fills by considering that

  - The fill-fill spacing distance is fixed and cannot be varied
  - The fill width is fixed
  - The fill-routing spacing can be adjusted by the tool (until you specify the minimum value in the parameter or options file) if the placed virtual metal fill fits irregularly within the space to be filled

- `adjustable_fill_fill`

  Places virtual metal fills by considering that

Feedback

  - The fill-routing distance is fixed and cannot be varied
  - The fill width is preferentially fixed
  - The fill-fill spacing can be varied (until you specify the minimum value in the parameter or options file) if the placed virtual metal fill pattern fits irregularly within the space to be filled

  If only one polygon can fit in the space between two routing signal nets, the tool places only one virtual metal fill polygon with a larger width to honor the rule of fixed fill-routing distance.

- `adjustable_fill_width`

  Places virtual metal fills by considering that

  - The fill-fill and fill-routing spacings are fixed
  - The virtual metal polygon width can be varied by the tool (until you specify the minimum value in the parameter or options file) if the virtual metal fill pattern fits irregularly within the space to be filled

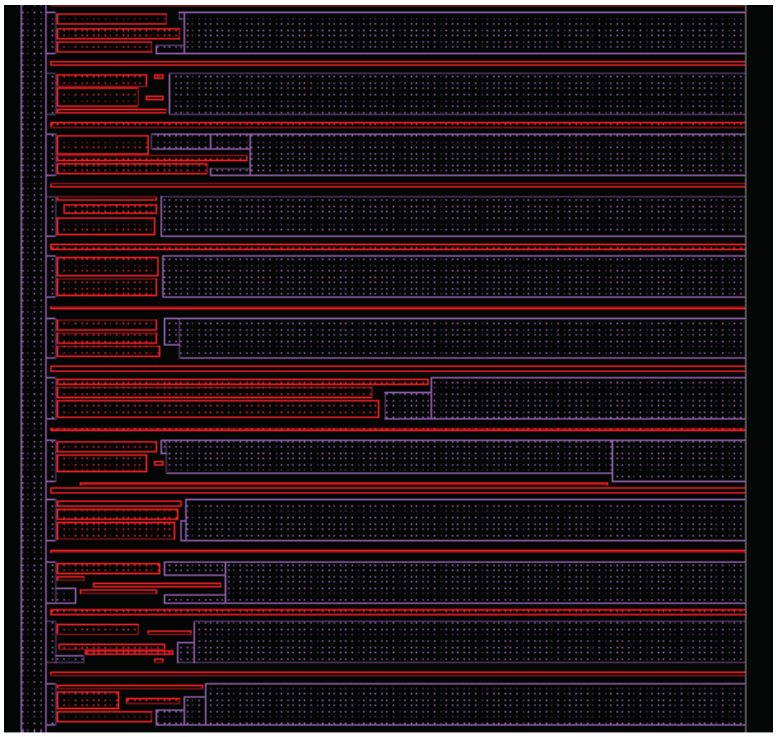
- `sheet`

  Places fill using the blanket sheet mode. Where,

  - Regions on which fill can be placed are considered as large blocks of fill material
  - The user-defined minimum fill-routing distance is always preserved and fill-fill spacing is not considered

To avoid polygon fragmentation, set the `min_fill_length` statement to `WMIN`. Figure 230 depicts zones where virtual metal fill polygons are placed using the sheet algorithm.

Figure 230 Virtual Metal Fill Polygons Surrounded With Design Polygons



See Also

- VIRTUAL_METAL_FILL_PARAMETER_FILE
- VIRTUAL_METAL_FILL_PARAMETERIZE
- VIRTUAL_METAL_FILL_POLYGON_HANDLING
- Virtual Metal Fill

Feedback

VIRTUAL_METAL_FILL_PARAMETER_FILE

Specifies a file containing parameters that model virtual metal fill.

Syntax

```
VIRTUAL_METAL_FILL_PARAMETER_FILE: vmf_file1 [vmf_file2 ...]
```

Arguments

Argument	Description
vmf_file1, vmf_file2 ...	One or more metal fill parameter file names Default: none

Description

Before the insertion of real metal fill, you can insert virtual metal fill polygons for better estimation of parasitics and timing early in the design process. Virtual metal fill emulates track-fill style metal fill using large aspect ratio polygons.

Caution:

Virtual metal fill is not intended for use in a signoff flow because it does not represent the actual fill in a design.

The `VIRTUAL_METAL_FILL_PARAMETER_FILE` command specifies a file that contains parameters to define the virtual metal fill polygon geometry and placement. You must also use the `VIRTUAL_METAL_FILL_POLYGON_HANDLING` command when you provide a virtual metal fill parameter file.

The StarRC tool accepts the following file types for the virtual metal fill parameter file:

- A metal fill parameter file generated by Fusion Compiler or IC Compiler II in the fill run directory `track_fill_params.rh`, which is recognized by the presence of the layer mapping definition. You can specify multiple metal fill runset files in the argument list of the `VIRTUAL_METAL_FILL_PARAMETER_FILE` command, in which case the first file in the list must contain the layer mapping information.
- An ASCII file generated by the Fusion Compiler or IC Compiler II tool with the `set_extraction_options -virtual_metalfill_parameter_file` command. You can specify only one of these files in the argument list.
- A file generated by the StarRC tool by setting the `VIRTUAL_METAL_FILL_PARAMETERIZE` command to `YES`. Use this command to analyze real metal fill and write a parameter file based on its layout. You can analyze a metal

fill design strategy one time, then use the generated parameter file for subsequent extraction runs for the same technology.

- A manually-created ASCII file. You can specify only one of these files in the argument list.

A manually-created ASCII file must contain one line for each database layer for which to create virtual metal fill.

To skip parameter generation for a design layer, provide a line in the parameter file that contains only the layer name and the keyword `S` for routing direction.

Otherwise, each line must use the following syntax:

```
layer_name direction
           fill_width min_fill_length     max_fill_length      \
           min_fill_route_w_spacing       fill_route_l_spacing \
           min_fill_fill_w_spacing        fill_fill_l_spacing  \

           min_fill_blockage_w_spacing    fill_blockage_w_spacing  \
           min_fill_ndr_w_spacing         fill_ndr_l_spacing    \
           fill_chip_w_spacing            fill_chip_l_spacing    \
           fill_blockage_excluded_cells
```

The `min_fill_ndr_w_spacing` and `fill_ndr_l_spacing` parameters are used only for nondefault rule (NDR) nets specified with the `VIRTUAL_METAL_FILL_NDR_NETS` command.

The StarRC tool checks the validity of the parameters and issues warning or error messages as needed.Table 95 describes the parameters. The last 7 parameters have defaults. You can omit the last 1, 3, 5, or 7 parameters at the end of a line to use the defaults. Some parameters must be specified in pairs, as noted in the table.

Note:

All dimensions in the virtual metal fill parameter file must be in nanometers. Dimensions should be the scaled size if a half-node scale factor is in use.

Table 95 Virtual Metal Fill Parameter File Values

Parameter	Type	Valid values	Description
`layer_name`	string	Any valid database layer name	A layer for which to create virtual me tal fill
`direction`	character	`V` (vertical) `H` (horizontal) `U` (unknown) `S` (skip)	Routing direction of virtual metal fill shapes

Feedback

Table 95 Virtual Metal Fill Parameter File Values (Continued)

Parameter	Type	Valid values	Description
`fill_width`	float	Greater than or equal to the WMIN value for the layer	Width of the virtual metal fill shapes in the direction perpendicular to the routing direction
`min_fill_length`	float	Greater than or equal to the WMIN value for the layer	Minimum length of the virtual metal fill shapes in the direction parallel to the routing direction
`max_fill_length`	float	Greater than or equal to the `min_fill_length` value	Maximum length of the virtual metal fill shapes in the direction parallel to the routing direction
`min_fill_route_ w_spacing`	float	Greater than or equal to the SMIN value for the layer	Minimum spacing between virtual fill shapes and design shapes in the direction perpendicular to the routing direction
`fill_route_ l_spacing`	float	Greater than or equal to the SMIN value for the layer	Spacing between virtual fill shapes and design shapes in the direction parallel to the routing direction
`min_fill_fill_ w_ spacing`	float	Greater than or equal to the SMIN value for the layer	Minimum spacing between virtual fill shapes in the direction perpendicular to the routing direction
`fill_fill_ l_spacing`	float	Greater than or equal to the SMIN value for the layer	Spacing between virtual fill shapes in the direction parallel to the routing direction
`min_fill_blocka ge_ w_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `min_fill_route_w_spacing`. Must specify along with `fill_blockage_l_spacing`.	Minimum spacing between virtual fill shapes and design blockage shapes in the direction perpendicular to the routing direction
`fill_blockage_ l_ spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `fill_route_l_spacing`. Must specify along with `min_fill_blockage_w_spaci ng`.	Spacing between virtual fill shapes and design blockage shapes in the direction parallel to the routing direction

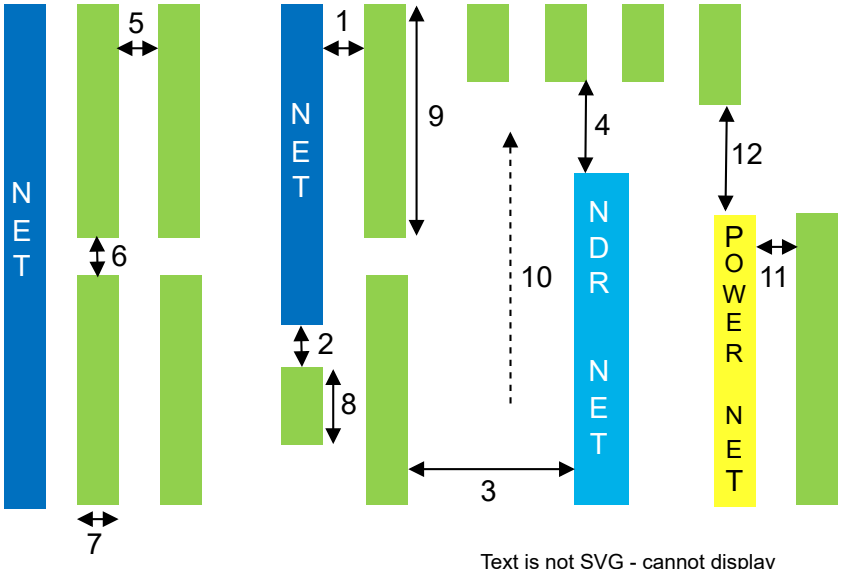
Feedback

Table 95 Virtual Metal Fill Parameter File Values (Continued)

Parameter	Type	Valid values	Description
`min_fill_ndr_ w_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `min_fill_route_w_spacing`. Must specify along with `fill_ndr_l_spacing`.	Minimum spacing between virtual fill shapes and NDR nets in the direction perpendicular to the routing direction
`fill_ndr_ l_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `fill_route_l_spacing`. Must specify along with `min_fill_ndr_w_spacing`.	Spacing between virtual fill shapes and NDR nets in the direction parallel to the routing direction
`fill_chip_ w_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to 0. Must specify along with `fill_chip_l_spacing`.	Minimum spacing between virtual fill shapes and the chip boundary in the direction perpendicular to the routing direction
`fill_chip_ l_spacing`	float	Greater than or equal to the SMIN value for the layer. Defaults to the value of `fill_route_l_spacing`. Must specify along `with fill_chip_w_spacing`.	Spacing between virtual fill shapes and the chip boundary in the direction parallel to the routing direction
`fill_blockage_ excluded_cells`	float	Greater than or equal to the SMIN value for the layer. Defaults to 200 nm.	Spacing between virtual fill shapes and excluded cells

Feedback

Figure 231 Virtual Metal Fill Parameters



1. min_fill_route_w_spac...
2. fill_route_l_spacing
3. min_fill_ndr_w_spacing
4. fill_ndr_l_spacing
5. min_fill_fill_w_spaci...
6. fill_fill_l_spacing
7. fill_width
8. min_fill_length
9. max_fill_length
10. direction (V)
11. fill_pwr_w_ spacing
12. fill_pwr_l_ spacing

Text is not SVG - cannot display

See Also

- VIRTUAL_METAL_FILL_NDR_NETS
- VIRTUAL_METAL_FILL_POLYGON_HANDLING
- VIRTUAL_METAL_FILL_PARAMETERIZE
- VIRTUAL_METAL_FILL_EXCLUDED_CELLS
- Virtual Metal Fill

Feedback

VIRTUAL_METAL_FILL_PARAMETERIZE

Allows you to create a parameter file from a design with real metal fill in the same design technology.

Syntax

```
VIRTUAL_METAL_FILL_PARAMETERIZE: YES | NO
```

Arguments

Argument	Description
NO (default)	Does not analyze real metal fill or create a parameter file
YES	Creates a parameter file based on the layout of real metal fill in the design

Description

Before the insertion of real metal fill, you can insert virtual metal fill polygons for better estimation of parasitics and timing early in the design process. Virtual metal fill emulates track-fill style metal fill using large aspect ratio polygons.

You can generate a parameter file based on the layout of real metal fill by setting the VIRTUAL_METAL_FILL_PARAMETERIZE command to YES. The StarRC tool analyzes real metal fill in the design, generates parameters based on the fill layout, and saves the parameter file to the name specified in the VIRTUAL_METAL_FILL_PARAMETER_FILE command.

You can analyze a metal fill design strategy one time, then use the generated parameter file for subsequent extraction runs for the same technology.

See Also

- VIRTUAL_METAL_FILL_POLYGON_HANDLING
- VIRTUAL_METAL_FILL_PARAMETER_FILE
- VIRTUAL_METAL_FILL_EXCLUDED_CELLS
- Virtual Metal Fill

Feedback

VIRTUAL_METAL_FILL_POLYGON_HANDLING

Specifies the treatment of virtual metal fill polygons.

Syntax

```
VIRTUAL_METAL_FILL_POLYGON_HANDLING: IGNORE | GROUNDED | FLOATING
                                     | AUTOMATIC
```

Arguments

Argument	Description
`IGNORE` (default)	Ignores the effects of virtual metal fill
`GROUNDED`	Treats virtual metal fill polygons as grounded
`FLOATING`	Treats virtual metal fill polygons as floating

Description

Before the insertion of real metal fill, you can insert virtual metal fill polygons for better estimation of parasitics and timing early in the design process. Virtual metal fill emulates track-fill style metal fill using large aspect ratio polygons.

Caution:

Virtual metal fill is not intended for use in a signoff flow because it does not represent the actual fill in a design.

The `VIRTUAL_METAL_FILL_POLYGON_HANDLING` command specifies how to treat the metal fill; the command applies to both virtual and real fill.

You must also use the `VIRTUAL_METAL_FILL_PARAMETER_FILE` command to specify a parameter file to define the virtual metal fill polygon geometry and placement.

Examples

This command treats the virtual metal fill polygons as grounded:

```
VIRTUAL_METAL_FILL_POLYGON_HANDLING: GROUNDED
```

See Also

- VIRTUAL_METAL_FILL_NDR_NETS
- VIRTUAL_METAL_FILL_PARAMETER_FILE
- Virtual Metal Fill

Feedback

WIDE_DEVICE_TERM_RESISTANCE

Activates equipotential line node handling for resistor devices.

Syntax

```
WIDE_DEVICE_TERM_RESISTANCE: RES [res1 res2 ...]
```

Arguments

Argument	Description
NONE (default)	Treats resistor terminals as point nodes
RES	Treats resistor terminals as line nodes
[res1 res2 ...]	Specifies resistor device names with the RES option The command treats resistor terminal nodes of only specified device names as line nodes. Note that the device names that are not specified are treated as point nodes. No wildcards allowed.

Description

This command treats resistor terminal nodes as line nodes, as opposed to point nodes, which is the default behavior. With this treatment, the StarRC tool identifies the terminal or body boundary line and extracts parasitic resistance orthogonal to that line.

The WIDE_DEVICE_TERM_RESISTANCE command to specify resitor device names should follow the same naming conventions as the CALIBRE_LVS_DEVICE_TYPE_RES and ICV_LVS_DEVICE_TYPE_RES commands.

For a resistor, an electrical point node is a physical approximation that assumes all current is concentrated at a single point instead of being distributed along the width of the material. For a device whose width is small relative to its length, this approximation is appropriate. However, when the width of the device is large relative to its length, as shown in Figure 232, the effect of current distribution along the body boundary must be considered during parasitic resistance extraction.

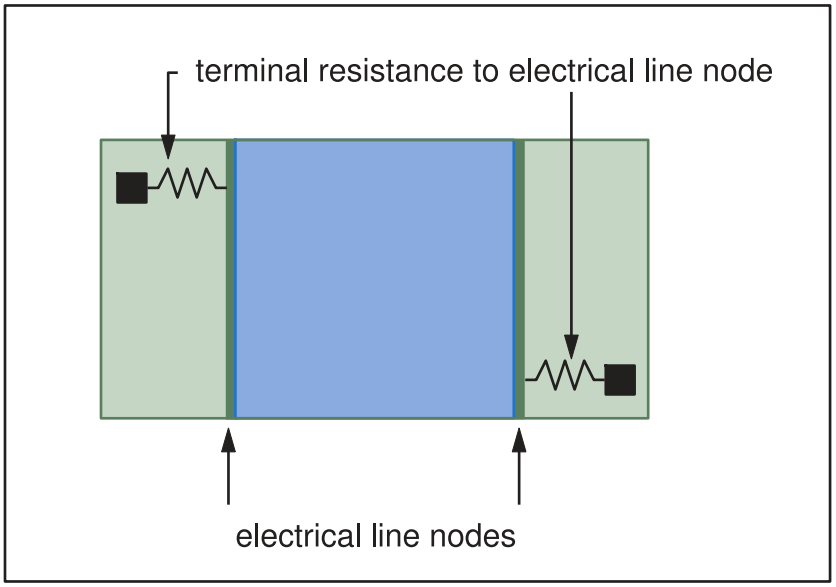
The StarRC tool first identifies the terminal layers for the resistors extracted by LVS tools. Because LVS tools always create an instance port on the abutting edge of the terminal shape and the resistor body shape, the StarRC tool can determine the current direction (perpendicular to the abutting edge) when extracting the resistance.

Feedback

The following usage notes apply:

- If the resistor terminal contains more than one shape, only the shape touched by the instance port is treated. The other shapes follow normal extraction processing.
- This command does not apply to bulk terminal layers of resistor devices.
- The resistor device terminals are assumed to be separated from interconnect layers.

Figure 232 Line Nodes for Resistor Terminals



See Also

- ICV_LVS_DEVICE_TYPE_RES
- CALIBRE_LVS_DEVICE_TYPE_RES

Feedback

XREF

Specifies the set of names used for netlist generation and analysis flows.

Syntax

```
XREF: NO | YES | COMPLETE
```

Arguments

Argument	Description
NO (default)	Reports the layout net names generated by the LVS tool during ideal layout extraction. These layout names are either generated or derived from the application of text. The layout instance names can either be the original GDSII instance name (if the ASSIGN_PROPERTY construct in Hercules is used), or names generated by Hercules.
YES	Reports all layout nets and devices occurring in the ideal layout netlist using schematic, net, and instance names wherever possible. When nets or devices exist that were not successfully matched during LVS, layout names are used. When a successful match did occur during LVS, the StarRC tool uses schematic names. The StarRC tool handles composite device merging by using schematic instance names with modifiers attached whenever layout devices outnumber schematic devices within a particular composite device pairing.
COMPLETE	Valid only in the Hercules flow. Reports every SKIP_CELLS/device in the schematic. The netlist includes all instances. If you do not want this, make sure that any net selected with the NETS command is also included in the NETLIST_SELECT_NETS command. Parasitics are written only for nets that were successfully cross-referenced to schematic nets. Nets that do not cross-reference to a schematic net are treated as ideal connections. Schematic model names are reported for SKIP_CELLS and primitive devices. Internal nets of the SERIES or PATHS merged devices do not have *|NET sections; they are written ideally in the instance section. Layout property merging for XREF:COMPLETE applies only to standard SPICE properties. Nonstandard properties cannot be merged.

Description

A GDSII based device-level extraction database contains both layout names and cross-referenced schematic names if a layout versus schematic verification was performed. The XREF command specifies which set of names to provide to the StarRC tool for netlist creation and analysis flows. It also determines which devices and nets to retain.

See Also

- Cross-Referencing in Transistor-Level Flows

Feedback

XREF_FEEDTHRU_NETS

Specifies whether to output pure layout feedthrough nets and ports in the `XREF: COMPLETE` netlist.

Syntax

```
XREF_FEEDTHRU_NETS: YES | NO
```

Arguments

Argument	Description
`YES`	Generates pure layout feedthrough nets and ports in the `XREF:COMPLETE` netlist
`NO` (default)	Does not generate feedthrough nets and ports

Description

This command specifies whether to output pure layout feedthrough nets and ports in the `XREF: COMPLETE` netlist. These are routes that cross a hierarchical boundary but whose ports were excluded from LVS, because they have no correspondence in the schematic netlist.

This command has no effect on `XREF` arguments other than `COMPLETE`. Pure layout feedthrough nets are always included in the other `XREF` command modes, because the other modes are layout-based.

Note:

The `CREATE_PORTS` option must be enabled in the Hercules runset for `XREF_FEEDTHRU_NETS: YES`.

See Also

- XREF

Feedback

XREF_LAYOUT_INST_PREFIX

Specifies a prefix for layout device instance names that were not cross-referenced to a schematic device by the LVS tool.

Syntax

```
XREF_LAYOUT_INST_PREFIX: prefix
```

Arguments

Argument	Description
prefix	Prefix used for netlist generation Default: `ld_`

Description

This prefix is applicable only for layout-based `XREF` mode `YES`, which generates a netlist for every layout element whether or not it was cross-referenced. Device instances that have no LVS comparison information, such as filtered layout instances, are written in the netlist with this prefix followed by the layout instance name.

See Also

- XREF_LAYOUT_NET_PREFIX
- XREF

Feedback

XREF_LAYOUT_NET_PREFIX

Sets a prefix for layout net names that are not cross-referenced to a schematic net by the LVS tool.

Syntax

```
XREF_LAYOUT_NET_PREFIX: prefix
```

Arguments

Argument	Description
prefix	Prefix used for net names in netlist generation Default: `ln_`

Description

This prefix is applicable only for the layout-based `XREF` mode `YES`, which generates a netlist for every layout element whether or not it was cross-referenced. Nets that have no LVS comparison information, such as dangling and floating nets, are written with this prefix followed by the layout net name.

See Also

- XREF_LAYOUT_INST_PREFIX
- XREF

Feedback

XREF_SWAP_MOS_SD_PROPERTY

Specifies a pair of MOS properties to be swapped.

Syntax

```
XREF_SWAP_MOS_SD_PROPERTY: prop1 prop2 [model_name]
```

Arguments

Argument	Description
prop1 prop2	A pair of properties to be swapped
model_name	Specifies that prop1 and prop2 are swapped only for *model_name* device models

Description

This command specifies a pair of MOS properties to be swapped. The pair of properties has the same swapping behavior as generic MOS source and drain properties such as, ad, ps, and pd.

The StarRC tool automatically swaps the following properties:

```
(as, ad) (ps, pd) (nrs, nrd) (rsc, rdc)
(asej, adej) (psej, pdej) (aseo, adeo) (pseo, pdeo)
```

The `XREF_SWAP_MOS_SD_PROPERTY` statement affects SPICE, standard parasitic format, and any other netlist format that has instances with properties. It does not affect SPEF files, which do not include device parameters.

Examples

To restrict the property swapping to a specific device model, specify the model name in the `XREF_SWAP_MOS_SD_PROPERTY` statement as shown in the following example:

```
XREF_SWAP_MOS_SD_PROPERTY: as5 ad5 nch_mac5
```

See Also

- MODEL_TYPE
- XREF

Feedback

XREF_USE_LAYOUT_DEVICE_NAME

Specifies whether to use layout device names when the XREF command is set to YES.

Syntax

```
XREF_USE_LAYOUT_DEVICE_NAME: YES | NO
```

Arguments

Argument	Description
YES	Netlist layout model names for all devices with XREF: YES. A layout model name is a primary device model name used in a device extraction command in a Hercules, IC Validator, or Calibre rule file.
NO (default)	Netlist schematic model names for all devices with XREF: YES; continue to use layout model names for XREF: NO. A schematic model name is a device model name that occurs in a schematic netlist used for an LVS (Hercules flow) or that occurs in NETLIST MODEL or TEXT MODEL commands (Calibre flow).

Description

Device model names can originate from one of two places:

- Schematic model name from a Hercules flow or NETLIST MODEL rule-file command from a Calibre flow
- Layout model name specified in device extraction command

See Also

- XREF
- XREF_USE_LAYOUT_TERMINAL_NAME

Feedback

XREF_USE_LAYOUT_TERMINAL_NAME

Specifies which terminal names to use when the XREF command is set to YES.

Syntax

```
XREF_USE_LAYOUT_TERMINAL_NAME: YES | NO
```

Arguments

Argument	Description
YES	Use layout terminal names for all devices with XREF: YES. For the IC Validator flow, a layout terminal name is the terminal name defined in the "pins" statement in the device definition in the runset report file (specified with the ICV_RUNSET_REPORT_FILE command). For the Calibre Connectivity Interface flow, the terminal name is a predefined name from the StarRC tool.
NO (default)	Use schematic terminal names for all devices with XREF: YES. For the IC Validator flow, a schematic terminal name is the terminal name defined in the "schematic_devices" statement in the device definition in the runset report file (specified with the ICV_RUNSET_REPORT_FILE command). For the Calibre Connectivity Interface flow, the schematic terminal name is the terminal name from the Calibre device template definition in the layout netlist file or device table.

Description

Table 96 lists the terminal names for the Calibre Connectivity Interface flow:

- Flow A: Layout netlist file (.nl file) from a Calibre flow
- Flow B: StarRC tool predefined names. For user-defined devices, terminal names are T1, T2, T3, and so on.

Table 96 Calibre Flow Device Terminal Names

Device type	Flow A	Flow B
MOS	D G S [B]	DRAIN GATE SOURCE [BULK]
CAPACITOR	POS NEG	A B
BJT	C B E	COLL BASE EMIT
DIODE	POS NEG	ANODE CATHODE
INDUCTOR	POS NEG	A B

Feedback

Table 96 Calibre Flow Device Terminal Names (Continued)

Device type	Flow A	Flow B
RES	POS NEG	A B

Table 97 shows how the `XREF_USE_LAYOUT_TERMINAL_NAME` and `XREF_USE_LAYOUT_DEVICE_NAME` commands affect the terminal names.

Table 97 Effect of StarRC Commands on Terminal Names

Setting of XREF_USE_LAYOUT_TERMINAL_NAME	Names used for XREF_USE_LAYOUT_DEVICE_NAME: YES	Names used for XREF_USE_LAYOUT_DEVICE_NAME: NO
Default (not set)	Flow B	Flow A
YES	Flow B	Flow B
NO	Flow A	Flow A

See Also

- XREF
- XREF_USE_LAYOUT_DEVICE_NAME

Feedback

ZONE_COUPLE_TO_NET

Couples noncritical material outside a defined macro to the specified net.

Syntax

```
ZONE_COUPLE_TO_NET: net_name
```

Arguments

Argument	Description
net_name	The net name to the noncritical material outside the defined macro Default: none

Description

This command is analogous to the SKIP_CELLS_COUPLE_TO_NET command, except that it applies to overhead or adjacent material when you are doing the in-context extraction of a macro or using the RING_AROUND_THE_BLOCK in-context approximation.

You must specify the NETLIST_FORMAT: SPEF and COUPLE_TO_GROUND: NO commands to use this command.

See Also

- ZONE_COUPLE_TO_NET_LEVEL

Feedback

ZONE_COUPLE_TO_NET_LEVEL

Specifies whether to append the layer number of the material outside the macro.

Syntax

```
ZONE_COUPLE_TO_NET_LEVEL: YES | NO
```

Arguments

Argument	Description
YES	Appends the layer number of the material outside a macro.
NO (default)	Does not append the layer number of the material outside a macro.

Description

This command appends the ZONE_COUPLE_TO_NET name to the real layer number for the SKIP_CELLS material.

This command is ignored unless the ZONE_COUPLE_TO_NET command is specified.

See Also

- ZONE_COUPLE_TO_NET

15

ITF Statements

The Interconnect Technology Format (ITF) file defines a cross section profile of the process. The ITF file consists of an ordered list of conductor and dielectric layer definition statements. The layers are defined from the topmost dielectric layer to the bottommost dielectric layer, excluding the substrate, in a way that is consistent with the physical manufacturing process.

The following reference pages describe the ITF statements and their options. Use the ITF statements to create an ITF file with the following structure:

```
TECHNOLOGY = process_name
REFERENCE_DIRECTION = VERTICAL | HORIZONTAL | GATE
[GLOBAL_TEMPERATURE = temp_value]
[BACKGROUND_ER = value]
[HALF_NODE_SCALE_FACTOR = scale_factor]
[USE_SI_DENSITY = YES | NO ]
[DROP_FACTOR_LATERAL_SPACING = value]

DIELECTRIC top_dielectric_name {…}
CONDUCTOR top_conductor_name {…}
[…]
[DIELECTRIC bottom_dielectric_name{…}]

VIA top_via_name {…}
[…]
VIA bottom_via_name {…}
```

Nomenclature

The terms command, option, and statement are synonymous. The term block refers to a group of related statements. The term keyword most often refers to a syntax word that is used within a higher-level command or option.

Delimiters in Lists

In lists and matrixes, both commas and spaces are accepted as delimiters between values. Spaces are the preferred delimiters. Line feeds, carriage returns, tabs, and other nonprinting characters are ignored.

Feedback

Order of Keywords

Many commands (options) include a number of keywords, which might be either optional or required. Unless otherwise specified, the order of those keywords within a command or option does not matter.

Comments in an ITF File

In an ITF file, text on a line after a dollar sign ($) is a comment that is not interpreted. You can begin a comment anywhere in the line. Comment lines do not wrap.

Restrictions for Layer Names

Layer names must obey the following restrictions:

- Layer names are case-sensitive.
- Names must contain only alphanumeric characters and underscores (_) unless otherwise noted.
- Names must begin with an alphabetic character.

Feedback

AIR_GAP_VS_SPACING

Defines air gap parameters. Valid within a CONDUCTOR block for a routing layer.

Syntax

```
AIR_GAP_VS_SPACING {
    SPACINGS { s1 s2 .. sn }
    AIR_GAP_WIDTHS { w(s1) w(s2) … w(sn) }
    AIR_GAP_THICKNESSES { t(s1) t(s2) … t(sn) }
    AIR_GAP_BOTTOM_HEIGHTS { h(s1) h(s2) …  h(sn) }
}
```

Arguments

Argument	Description
s1 s2 … sn	Spacing between two conductors Units: microns
w(s1) w(s2) … w(sn)	Width of the air gap formed at the corresponding spacing value Units: microns
t(s1) t(s2) … t(sn)	Thickness of the air gap formed at the corresponding spacing value Units: microns
h(s1) h(s2) … h(sn)	Height of the bottom of the air gap from the bottom of the conductor at the corresponding spacing value Units: microns

Description

Use an AIR_GAP_VS_SPACING table to define the parameters of an air gap. The number of arguments in each row of the table must be equal. The smallest spacing value *s1* must be equal to the value of the SMIN keyword used in the CONDUCTOR statement for a routing layer.

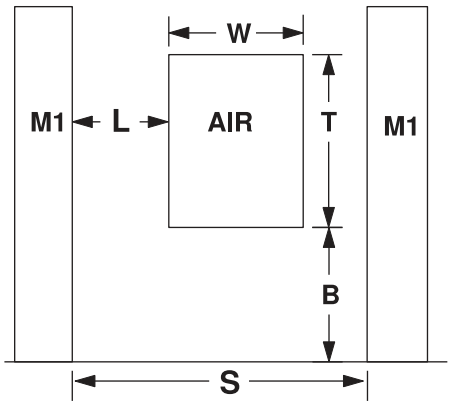
If the spacing between the polygons is greater than *sn*, an air gap does not form.

Lists of values are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Figure 233 illustrates the parameters used in the AIR_GAP_VS_SPACING command.

Feedback

Figure 233 Process Side View With Dielectric Air Gaps



S = spacing between neighboring lines (SPACINGS)
W = width of the air gap (AIR_GAP_WIDTHS)
T = thickness of the air gap formed (AIR_GAP_THICKNESSES)
B = height of the bottom of the air gap from the bottom of metal (AIR_GAP_BOTTOM_HEIGHTS
L = spacing between the metal and air gap ((S-W) /2)

Examples

```
AIR_GAP_VS_SPACING {
     SPACINGS { 0.3  0.5  0.7  1.0  3.0  }
     AIR_GAP_WIDTHS { 0.1 0.09 0.09 0.15 0.20 }
     AIR_GAP_THICKNESSES    { 0.2  0.23 0.25 0.26 0.28 }
     AIR_GAP_BOTTOM_HEIGHTS { 0.1  0.14 0.18 0.20 0.22 }
}
```

See Also

- CONDUCTOR

Feedback

AREA

Specifies the default area of a via. Valid within a VIA or TSV block.

Syntax

```
AREA = via_area
```

Arguments

Argument	Description
via_area	Area of default via Units: square microns Default: 1.0e-6

Description

Within a VIA block, the resistive properties of a standard via layer (not a trench contact via layer) must be specified with one of three mutually exclusive methods:

- Using the RHO keyword
- Using the RPV and AREA keywords (these keywords cannot be used alone in a VIA block)
- Using an RPV_VS_AREA table to specify resistance values for different via areas

Within a TSV block, the AREA keyword is a required keyword.

Examples

```
VIA via1 { FROM=m1 TO=m2 AREA=4 RPV=0.25 }
```

See Also

- RHO
- RPV
- RPV_VS_AREA
- VIA
- TSV

Feedback

ASSOCIATED_CONDUCTOR

Names another layer whose properties are related to the current layer. Valid within a DIELECTRIC block or a CONDUCTOR block.

Syntax

```
ASSOCIATED_CONDUCTOR = layer_name
```

Arguments

Argument	Description
layer_name	Associated conductor layer name

Description

The usages of the ASSOCIATED_CONDUCTOR keyword are as follows:

- In a DIELECTRIC block, to name a conductor layer to which the dielectric layer conforms
- In a CONDUCTOR block, to name a layer to use for trench contact extensions
- In a CONDUCTOR block, to name a conductor layer whose thickness is a reference for the thickness of the current layer

In a DIELECTRIC block for a conformal dielectric layer, the ASSOCIATED_CONDUCTOR keyword names the layer to which the dielectric layer conforms. The following usage notes apply:

- Only one ASSOCIATED_CONDUCTOR keyword is allowed within a DIELECTRIC block.
- The ASSOCIATED_CONDUCTOR keyword can only be used with the IS_CONFORMAL keyword. However, the IS_CONFORMAL keyword can be used without the ASSOCIATED_CONDUCTOR keyword. When no ASSOCIATED_CONDUCTOR keyword is specified for an IS_CONFORMAL layer, the default is to measure from the top layer.
- Using the IS_CONFORMAL and ASSOCIATED_CONDUCTOR keywords is mutually exclusive with using the MEASURED_FROM keyword within the same DIELECTRIC block.
- When an ASSOCIATED_CONDUCTOR layer drops because of a DROP_FACTOR defined for a layer below it, the related IS_CONFORMAL dielectric layers also drop.
- The conductor named in the ASSOCIATED_CONDUCTOR keyword cannot be higher than the dielectric named in the IS_CONFORMAL keyword.
- If a conductor above a conformal dielectric layer overlaps with the dielectric layer top wall thickness, the conductor cuts into the dielectric layer.

Feedback

In a CONDUCTOR block for a trench contact layer, the ASSOCIATED_CONDUCTOR keyword names a layer to use for the trench contact extensions. The following usage notes apply:

- In multigate models, a trench contact is created only for a conductor layer that has the LAYER_TYPE keyword set to TRENCH_CONTACT and connects to the diffusion layer in the GATE_DIFFUSION_LAYER_PAIR list in the MULTIGATE block.
- If the trench contact conductor block has an ASSOCIATED_CONDUCTOR keyword, the named associated layer is used for the trench contact extension.
- If the ASSOCIATED_CONDUCTOR keyword is missing, the trench contact extension is created using the same layer as the trench contact.

In a CONDUCTOR block for a metal routing layer, the ASSOCIATED_CONDUCTOR keyword names a layer to use as a thickness reference. The following usage notes apply:

- The THICKNESS_VARIATION_VS_MASK command specifies a layer thickness relative to another layer thickness. The two conductor layers must be linked by reciprocal ASSOCIATED_CONDUCTOR statements.
- The bottom heights of the two associated conductor layers must be identical. By definition, the two layers are covertical, which means they overlap in the vertical dimension.
- Interlayer dielectric (ILD) layer variations are not allowed for either of the two associated conductors, because ILD variations cause the conductor bottom heights to be different.
- You can use different thickness variation tables for the two associated conductors.
- The THICKNESS_VARIATION_VS_MASK table must be the only thickness variation specification for that layer.

Examples

```
DIELECTRIC D1 {
     IS_CONFORMAL
     ASSOCIATED_CONDUCTOR=met1
     SW_T=0.1 TW_T=0.1 ER=2.5
}
```

See Also

- IS_CONFORMAL
- DIELECTRIC
- LAYER_TYPE

Feedback

- CONDUCTOR
- THICKNESS_VARIATION_VS_MASK

Feedback

BACKGROUND_ER

Specifies the relative permittivity (dielectric constant) of the background dielectric.

Syntax

```
BACKGROUND_ER = relative_permittivity
```

Arguments

Argument	Description
relative_permittivity	Relative permittivity Default: 1.0

Description

The BACKGROUND_ER statement is an optional statement that can be included in the global parameters section of the ITF file. If the BACKGROUND_ER statement is not specified, the relative permittivity of the background dielectric is set to 1.0, the relative permittivity of air.

The background dielectric globally fills the cross section to an infinite height, effectively replacing air as the operating medium for the chip.

Relative permittivity settings within individual DIELECTRIC blocks override the global background value.

Examples

```
TECHNOLOGY = example_tech
GLOBAL_TEMPERATURE = 31.0
BACKGROUND_ER = 4.1
```

Feedback

BOTTOM_DIELECTRIC_ER

Specifies the relative permittivity of the dielectric region below a conductor. Valid within a `CONDUCTOR` block.

Syntax

```
BOTTOM_DIELECTRIC_ER = permittivity
```

Arguments

Argument	Description
permittivity	Relative permittivity of the dielectric

Description

The `BOTTOM_DIELECTRIC_ER` keyword must be specified along with the `BOTTOM_DIELECTRIC_THICKNESS` keyword within a `CONDUCTOR` block. If the specified conductor layer does not appear in a certain model, the bottom dielectric layer does not appear in the model either.

Examples

```
BOTTOM_DIELECTRIC_ER = 4.0
```

See Also

- BOTTOM_DIELECTRIC_THICKNESS
- CONDUCTOR

Feedback

BOTTOM_DIELECTRIC_THICKNESS

Specifies the thickness of the dielectric region below a conductor. Valid within a `CONDUCTOR` block.

Syntax

```
BOTTOM_DIELECTRIC_THICKNESS = thickness
```

Arguments

Argument	Description
thickness	Thickness of the dielectric Units: microns

Description

The `BOTTOM_DIELECTRIC_THICKNESS` keyword must be specified along with the `BOTTOM_DIELECTRIC_ER` option within a `CONDUCTOR` statement. If the specified conductor layer does not appear in a certain model, the bottom dielectric layer does not appear in the model either.

If a conductor with a bottom dielectric also has conformal dielectrics on the sides of the conductor, the side conformal dielectric layers extend to the lowest surface of the bottom conformal dielectric, as shown in Figure 234. When placing a conductor with bottom dielectric in the dielectric stack, the lowest surface of the bottom dielectric layer sits on the top surface of the planar dielectric layer defined below the conductor.

To specify the thickness of the conductor with the bottom dielectric layer, use the `THICKNESS` option in the `CONDUCTOR` statement. The top of a conductor that includes a bottom dielectric is placed above the planar dielectric at a height equal to the sum of the conductor thickness and the bottom dielectric thickness.

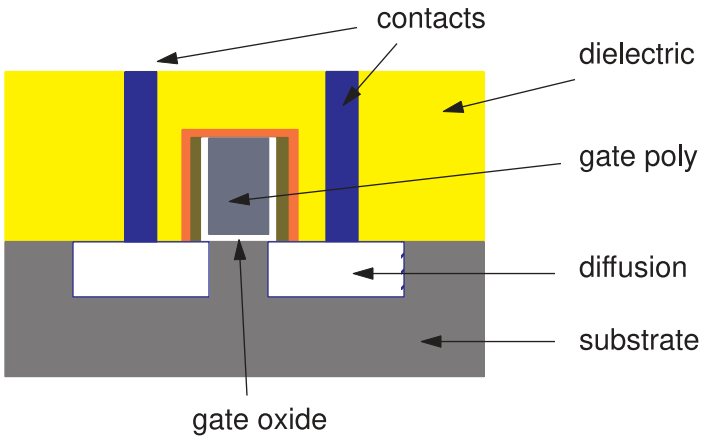
Examples

High-Permittivity Gate Oxide

The following statements model the high-permittivity dielectric under the gate shown in Figure 234:

```
CONDUCTOR gpoly {
  THICKNESS = 0.06
  WMIN = 0.03
  SMIN = 0.03
  BOTTOM_DIELECTRIC_THICKNESS = 0.002
  BOTTOM_DIELECTRIC_ER = 10.0
}
```

Figure 234 Modeling High-Permittivity Dielectric Under the Gate

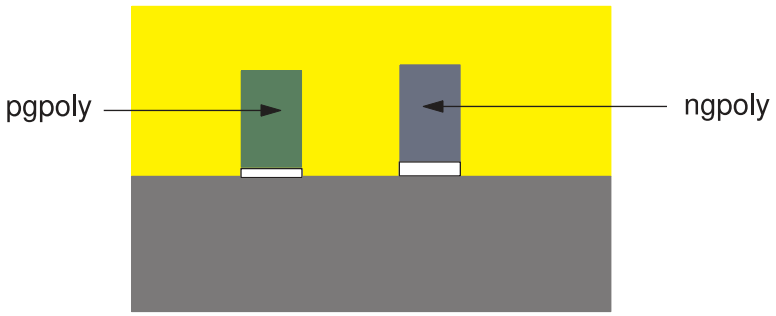


Independent Bottom Dielectric Regions in Covertical Conducting Layers

You can define independent bottom dielectric regions in covertical conducting layers. For example, the following statements model pgpoly and ngpoly conductors with different bottom dielectric regions, as shown in Figure 235:

```
CONDUCTOR pgpoly {
  THICKNESS = 0.06 WMIN = 0.03 SMIN = 0.03
  BOTTOM_DIELECTRIC_THICKNESS = 0.002
  BOTTOM_DIELECTRIC_ER = 10.0
}
CONDUCTOR ngpoly {
  THICKNESS = 0.06 WMIN = 0.03 SMIN = 0.03
  BOTTOM_DIELECTRIC_THICKNESS = 0.004
  BOTTOM_DIELECTRIC_ER = 12.0
}
```

Figure 235 Bottom Dielectric Layer With Covertical Layers



Feedback

See Also

- BOTTOM_DIELECTRIC_ER
- CONDUCTOR

Feedback

BOTTOM_THICKNESS_VS_SI_WIDTH

Specifies the bottom thickness of a conductor layer at different widths. Valid within a CONDUCTOR block.

Syntax

```
BOTTOM_THICKNESS_VS_SI_WIDTH [RESISTIVE_ONLY | CAPACITIVE_ONLY]
     { (s1, r1) (s2, r2) … (sn, rn) }
```

Arguments

Argument	Description
RESISTIVE_ONLY	Applies thickness adjustment to resistance only
CAPACITIVE_ONLY	Applies thickness adjustment to capacitance only
s1 … sn	Silicon widths in ascending order. The first entry should be the smallest possible silicon width of the layer coming from the drawn WMIN value.
r1 … rn	Relative changes in bottom thickness. This is the absolute thickness change from the bottom divided by the nominal thickness of the layer. The value is positive if the thickness becomes larger than the nominal value. The value is negative if the thickness becomes smaller than the nominal value.

Description

The BOTTOM_THICKNESS_VS_SI_WIDTH option models the bottom thickness of a conductor.

In a damascene process, deposited metal fills previously etched trenches. The variation in the trench etch depth affects the thickness of the interconnect as well as the vertical distance between metal interconnects. Both parasitic resistance and capacitance can be affected by these variations.

The grdgenxo tool does not check the validity of the silicon width values. The StarRC tool performs linear interpolation of the thickness for wires whose widths fall between entries in the table. The tool does not extrapolate beyond the table limits.

When the StarRC tool modifies a conductor thickness based on the BOTTOM_THICKNESS_VS_SI_WIDTH statement, the tool modifies the conductor sheet resistance values accordingly. The modifications apply to values specified in the RPSQ, RPSQ_VS_SI_WIDTH, and RPSQ_VS_WIDTH_AND_SPACING statements.

The following limitations apply:

- The grdgenxo tool automatically processes trapezoidal conductor cross sections. This means that at a given thickness coming from a change at the bottom or top, the specified ETCH_VS_WIDTH_AND_SPACING, ETCH_FROM_TOP, or SIDE_TANGENT value is automatically applied for the whole cross section when calculating the sensitivity.
- You can specify the BOTTOM_THICKNESS_VS_SI_WIDTH option along with the thickness variation from the top of the conductor by using the following options: THICKNESS_VS_DENSITY, THICKNESS_VS_WIDTH_AND_SPACING, POLYNOMIAL_BASED_THICKNESS_VARIATION.
- The BOTTOM_THICKNESS_VS_SI_WIDTH option cannot be specified in the same CONDUCTOR statement as the MEASURED_FROM option.
- The StarRC tool allows the BOTTOM_THICKNESS_VS_SI_WIDTH option to be used with multiple ETCH_VS_WIDTH_AND_SPACING tables. However, this combination might result in many sources of change for a metal line, making the behavior difficult to understand. This combination of ITF statements is not recommended.

Effective Thickness Calculation

The effective thickness is calculated as follows:

$$T = Tnom \times (1 + RTf(Deff) + RTf(W,S) + RTf(SiW))$$

where

- Tnom is the nominal thickness specified in the ITF file.
- RTf(Deff) is the relative thickness change due to density.
- RTf(W,S) is the relative change in thickness due to width and spacing.
- RTf(SiW) is the relative change in thickness due to silicon width.

The resistance and capacitance are computed after the effective thickness is computed.

Errors

An error occurs if the BOTTOM_THICKNESS_VS_SI_WIDTH option changes the relative thickness by more than 50 percent because the accuracy is compromised if the thickness changes greatly.

An error occurs if either the RESISTIVE_ONLY or CAPACITIVE_ONLY option is used with another BOTTOM_THICKNESS_VS_SI_WIDTH option without any qualifiers. The tool expects either a common BOTTOM_THICKNESS_VS_SI_WIDTH table or separate tables for the RESISTIVE_ONLY and CAPACITIVE_ONLY scenarios.

Feedback

See Also

- CONDUCTOR
- RPSQ
- RPSQ_VS_SI_WIDTH
- RPSQ_VS_SI_WIDTH_AND_LENGTH
- RPSQ_VS_WIDTH_AND_SPACING
- ETCH_VS_WIDTH_AND_SPACING
- SIDE_TANGENT
- THICKNESS_VS_DENSITY
- THICKNESS_VS_WIDTH_AND_SPACING
- POLYNOMIAL_BASED_THICKNESS_VARIATION
- MEASURED_FROM

Feedback

BW_T

Specifies the extension distance (bottom wall thickness) of the conformal dielectric below the conductor. Valid within a DIELECTRIC block.

Syntax

```
BW_T = thickness
```

Arguments

Argument	Description
thickness	The bottom conformal dielectric thickness Units: microns Default: thickness of the dielectric

Description

The BW_T option works with the RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER option, which specifies the dielectric constant of the associated silicon dielectric area between the raised diffusion and the gate processes. The BW_T value specifies the extension distance of the conformal dielectric below the conductor.

You can associate multiple conformal dielectric layers with BW_T values specified for the same conductor to produce multiple bottom conformal dielectrics. The BW_T thicknesses of multiple conformal dielectrics are additive. The BW_T dielectric listed first in the ITF file is the topmost BW_T dielectric associated with a particular conductor, that is, the closest bottom conformal dielectric to the conductor.

Use the following guidelines for the BW_T option:

- The BW_T keyword can only be used in dielectric statements that also contain IS_CONFORMAL and ASSOCIATED_CONDUCTOR keywords.
- The DAMAGE_THICKNESS and DAMAGE_ER keywords cannot be simultaneously specified with the BW_T keyword.
- The BOTTOM_DIELECTRIC_THICKNESS and BOTTOM_DIELECTRIC_ER keywords cannot be specified in a conductor that is associated with a conformal layer that has a BW_T keyword.
- For conformal layers without a BW_T specification, set the BW_T value to zero to ensure backward compatibility with existing ITF files.
- If you define a BW_T value for a dielectric layer, you must set the SW_T and TW_T values to zero.

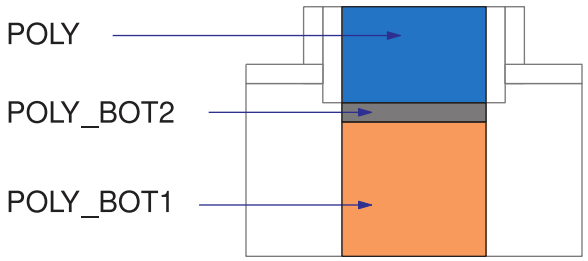
Feedback

Examples

The following example uses the BW_T keyword to define the bottom dielectrics shown in Figure 236.

```
$ Gate oxide bottom conformal dielectric (closest to poly)
DIELECTRIC POLY_BOT2 {
    THICKNESS=0.0 IS_CONFORMAL ASSOCIATED_CONDUCTOR=POLY
    SW_T=0 TW_T=0 BW_T=0.005 }
$ Silicon dielectric under gate oxide (farthest from poly)
DIELECTRIC POLY_BOT1 {
    THICKNESS=0.0 IS_CONFORMAL ASSOCIATED_CONDUCTOR=POLY
    SW_T=0 TW_T=0 BW_T=0.3 …}
```

Figure 236 Example of BW_T Option Usage



See Also

- DIELECTRIC
- MEASURED_FROM
- SW_T
- TW_T
- RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER
- THICKNESS

Feedback

CAPACITIVE_ONLY_ETCH

Identical to the ETCH option, except that only capacitance is affected. Valid within a CONDUCTOR block.

Syntax

```
CAPACITIVE_ONLY_ETCH = etch_value
```

Arguments

Argument	Description
etch_value	Absolute width adjustment for one sidewall. A positive value shrinks the conductor; a negative value expands it. Units: microns Default: 0.0

Description

The CAPACITIVE_ONLY_ETCH option applies an etch value to the sidewalls of a conductor. A positive value denotes conductor shrink; a negative value denotes conductor expansion. The adjusted conductor width is equal to the drawn width minus twice the etch value.

Use this option instead of the ETCH option to specify that an etch operation is to be used only for capacitance calculations.

If you use one of the ETCH options in addition to one or more ETCH_VS_WIDTH_AND_SPACING tables, the ETCH_VS_WIDTH_AND_SPACING operations are applied first, followed by the ETCH operation.

This option is not the same as ETCH_VS_WIDTH_AND_SPACING CAPACITIVE_ONLY.

Examples

```
CONDUCTOR metal1 {
    CAPACITIVE_ONLY_ETCH = 0.05
    THICKNESS=0.66 WMIN=0.15 SMIN=0.15 RPSQ=0.078
}
```

See Also

- ETCH
- RESISTIVE_ONLY_ETCH
- ETCH_VS_WIDTH_AND_SPACING

Feedback

CONDUCTOR

Describes the properties of a conductor layer.

Syntax

```
CONDUCTOR conductor_name {
  SMIN = min_spacing
  WMIN = min_width
  THICKNESS = cond_thk
  [LINKED_TO = layer_name]
  [LAYER_TYPE = GATE | FIELD_POLY | DIFFUSION | TRENCH_CONTACT | BUMP
     | ROUTING_VIA | TALL_CONTACT | CAPACITOR | RESISTOR]
  AIR_GAP_DIELECTRIC = k
  AIR_GAP_WIDTH = w
  AIR_GAP_THICKNESSES = T
  AIR_GAP_BOTTOM_HEIGHT = H
  AIR_GAP_SIDE_TANGENT = t
  [TC_ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY {…}]
  [DUAL_POLY = top_poly_name]
  [DEVICE_TYPE { … }]
  [T0 = nominal_temp
     | T0 = nominal_temp CRT1 = lin_coeff
     | T0 = nominal_temp CRT2 = quad_coeff
     | T0 = nominal_temp CRT1 = lin_coeff CRT2 = quad_coeff
     | T0 = nominal_temp CRT_VS_SI_WIDTH { … }]
  [ASSOCIATED_CONDUCTOR = associated_layer]
  [EXTENSIONMIN = min_extension]
  [SIDE_TANGENT = tan_value |(coco_tangent, poco_tangent)]
  [AIR_GAP_VS_SPACING {...} ]
  [DIELECTRIC_FILL_VS_SI_SPACING {...} ]
  [DIELECTRIC_FILL_EMULATION_VS_SI_SPACING {...} ]
  [BOTTOM_DIELECTRIC_THICKNESS = b_diel_thk
     BOTTOM_DIELECTRIC_ER = b_diel_er]
  [BOTTOM_THICKNESS_VS_SI_WIDTH … { … }]
  [DENSITY_BOX_WEIGHTING_FACTOR { … }]
  [THICKNESS_VS_DENSITY { … }
     | THICKNESS_VS_DENSITY_AND_WIDTH { … }
     | THICKNESS_VS_WIDTH_AND_SPACING { … }
     | THICKNESS_VARIATION_VS_MASK { … }
     | POLYNOMIAL_BASED_THICKNESS_VARIATION { … }]
  [DROP_FACTOR = value]
  [ETCH = value
     | CAPACITIVE_ONLY_ETCH = value
     | RESISTIVE_ONLY_ETCH = value]
  [ETCH_VS_WIDTH_AND_SPACING { … }]
  [FILL_RATIO = fill_ratio_value
     FILL_WIDTH = fill_width_value
     FILL_SPACING = fill_spacing_value
     FILL_TYPE = GROUNDED | FLOATING ]
  [GATE_TO_CONTACT_SMIN = gc_smin_value]
  [GATE_TO_DIFFUSION_CAP { … }]
```

Feedback

```
  [GATE_TO_DIFFUSION_ADJUSTMENT_CAP { … }]
  [GATE_PITCH {gpmin, ... gpmax}
   GATE_WIDTH {gwmin, ... gwmax}
   GATE_TO_CONTACT_SPACING {gcmin, ... gcmax}]
  [ILD_VS_WIDTH_AND_SPACING { … }]
  [IS_PLANAR]
  [MEASURED_FROM = dielectric_name | TOP_OF_CHIP]
  [RAISED_DIFFUSION_ETCH = rd_distance
     [RAISED_DIFFUSION_ETCH_TABLE { … }]
     RAISED_DIFFUSION_THICKNESS = rd_thickness
     RAISED_DIFFUSION_TO_GATE_SMIN = rd_spacing
     [RAISED_DIFFUSION_TO_GATE_SMIN_TABLE { … }]
     [RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER = rd_er]]
  [RPSQ = rpsq_value
       | RHO = rho_value
       | RPSQ_VS_SI_WIDTH { … }
       | RPSQ_VS_WIDTH_AND_SPACING { … }
       | RPSQ_VS_SI_WIDTH_AND_LENGTH { … }
       | RHO_VS_SI_WIDTH_AND_THICKNESS { … }
       | RHO_VS_WIDTH_AND_SPACING { … }]
  [SIDE_TANGENT = value
       | SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING { … }
       | SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING { … }]
  [USER_DEFINED_DIFFUSION_RESISTANCE … { … }]
  [VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH … { … }]
}
```

Arguments

Argument	Description
conductor_name	Name of the conductor layer
min_spacing	Minimum spacing between two geometries on this layer Units: microns
min_width	Minimum width of a geometry on this layer Units: microns
cond_thk	Thickness of the layer (minimum value is 0.001 micron) Units: microns
layer_name	Conductor layer that is capacitively equivalent; valid only for QTF flows
top_poly_name	For a dual polysilicon gate process, the name of the top polysilicon layer. This option is valid only when defining a bottom polysilicon layer with `LAYER_TYPE=GATE`.

Feedback

Argument	Description
nominal_temp	Nominal temperature Units: degrees Celsius Default: temperature specified by `GLOBAL_TEMPERATURE`
lin_coeff	Layer-specific linear temperature coefficient Default: 0
quad_coeff	Layer-specific quadratic temperature coefficient Default: 0
associated_layer	For trench contact layers only, the layer used for trench contact extensions
min_extension	Minimum allowable extension of the field poly layer beyond the gate polygon Units: microns
tan_value	Tangent of the contact sidewall angle. Allowed only for vias between the diffusion layer and the metal above the gate layer. Default: 0
coco_tangent	Tangent of the contact sidewall angle in the direction of contact-to-contact spacing. Allowed only for tall vias between the diffusion layer and the metal above the gate layer. Default: 0
poco_tangent	Tangent of the contact sidewall angle in the direction of poly-to-contact spacing. Allowed only for tall vias between the diffusion layer and the metal above the gate layer. Default: 0
b_diel_thk	Thickness of the bottom dielectric layer Units: microns
b_diel_er	Relative permittivity of the bottom dielectric layer
fill_ratio_value	Ratio of metal fill coverage
fill_width_value	Average width of metal fill objects Units: microns
fill_spacing_value	Average lateral spacing between signal nets and metal fill objects Units: microns
dielectric_name	Name of a dielectric layer whose top surface is a reference height
gc_smin_value	Gate to contact SMIN value Units: microns

Feedback

Argument	Description
gpmin, ... *gpmax*	Typical gate pitch values, in ascending order, for tall contact modeling. Must provide at least two values (minimum and maximum) but no more than three. Units: microns
gwmin, ... *gwmax*	Typical gate width values, in ascending order, for tall contact modeling. Must provide at least 2 values (minimum and maximum) but no more than 5. Units: microns
gcmin, ... *gcmax*	Typical gate to tall contact spacing values, in ascending order, for tall contact modeling. Must provide at least 2 values (minimum and maximum) but no more than 5. Units: microns
rpsq_value	Resistance per square of the conducting layer, based on the silicon width (physical width) Units: ohms/square Default: 0
rho_value	Bulk resistivity of the conductor layer Units: ohms-micron
k	A dielectric layer of the air gap Default: 1
w	Width of the air gap Units: microns
T	Thickness of the air gap Units: microns
H	Height of the bottom of the air gap from the top of the gate conductor layer Units: microns
t	Side tangent of the air gap Default: 0

Description

The CONDUCTOR statement describes the properties of a conductor layer such as minimum width, minimum spacing, thickness, resistivity, and process variation.

You can use a maximum of 50 CONDUCTOR statements in the ITF file.

Feedback

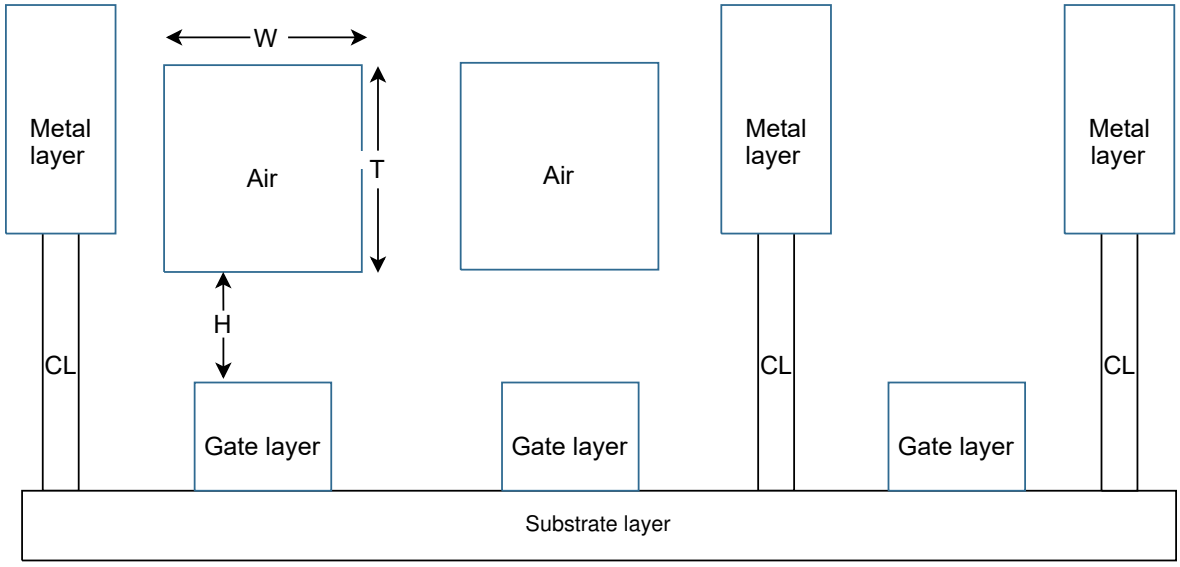
Specifying Air Gap Parameters for Gate Layers

Use the air gap parameters to define the parameters on gate layers. Valid within only a gate `CONDUCTOR` block. Air gap is created over the shapes of only those database layers, as shown in Figure 237, that are mapped to the gate conductor layer for which you have defined the air gap parameters in MAPPING_FILE.

Note:

You can specify the parameters only after you specify the layer type in a conductor block.

Figure 237 Modeling air space over gate layers



W is AIR_GAP_WIDTH
T is AIR_GAP_THICKNESSES
H is AIR_GAP_BOTTOM_HEIGHT

Air indicates air space over gate layers

CL indicates contact layer

Examples

The following example shows a simple `CONDUCTOR` statement for a metal layer.

```
CONDUCTOR M1 { THICKNESS=0.8 WMIN=0.5 SMIN=0.45 RPSQ=0.041 }
```

The following example shows how to define air gap parameters in a gate `CONDUCTOR` block

```
CONDUCTOR PS {
  …
  LAYER_TYPE = GATE
  …
  AIR_GAP_WIDTH = 0.13000
  AIR_GAP_THICKNESS = 0.500
```

```
  AIR_GAP_BOTTOM_HEIGHT = 0.19000
  AIR_GAP_DIELECTRIC = 1.00000
  AIR_GAP_SIDE_TANGENT = 0.00000
  …

}
```

See Also

- Conductor Layer Thickness Variation
- Bottom Conductor Thickness Variation
- Conductor Sheet Zones
- Table-Based Modeling of Tall Contacts
- Conductor Drop Factor
- Layer Etch
- Dual Polysilicon Gate Process

Feedback

CONTACT_TO_CONTACT_SPACING

Specifies contact spacing values for tall contact modeling. Valid within a VIA block.

Syntax

```
CONTACT_TO_CONTACT_SPACING {smin, s2, ..., smax}
```

Arguments

Argument	Description
smin, s2, ..., smax	Typical contact spacing values, in ascending order, for table-based tall contact modeling. Must provide at least 2 values (minimum and maximum) but no more than 5. Units: microns

Description

Tall contacts that are very large or that exhibit strong coupling capacitance to other tall contacts can be modeled by using a combination of options in the CONDUCTOR and VIA definitions in the ITF file. Figure 238 illustrates the tall contact parameters used in the definitions.

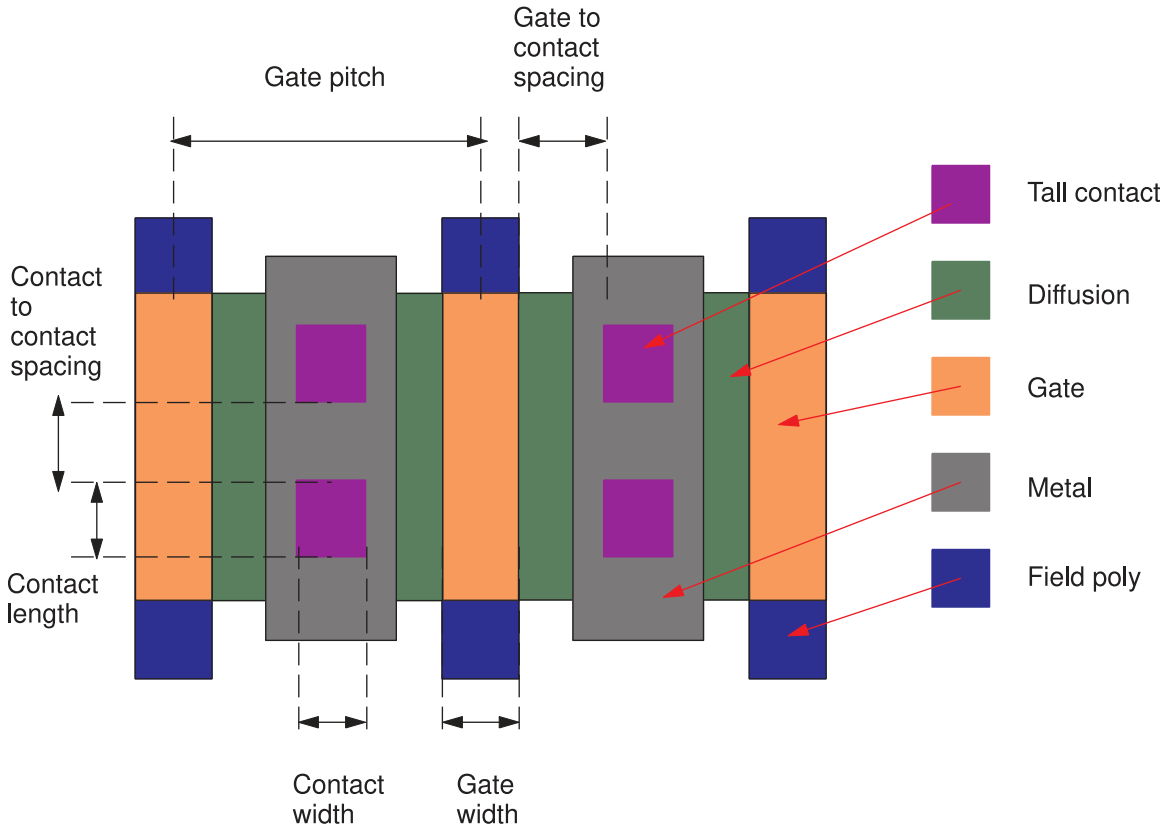
To model a tall contact with the table-based method, follow these steps:

- Include the following options in the VIA definition in the ITF file:
  - Set the LAYER_TYPE option to TALL_CONTACT.
  - (Optional) Set the DEVICE_TYPE option to a unique name. This name should be specified for one via layer and one gate conductor layer to indicate that they are used together in a tall contact structure.
  - (Optional) Include either or both of the CONTACT_WIDTH_AND_LENGTH and CONTACT_TO_CONTACT_SPACING options.
- Include the following options in the CONDUCTOR definition in the ITF file:
  - Set the LAYER_TYPE option to GATE.
  - (Optional) Set the DEVICE_TYPE option to the same name specified in the VIA definition.
  - (Optional) Include the GATE_PITCH, GATE_WIDTH, and GATE_TO_CONTACT_SPACING options.

For simultaneous multicorner (SMC) extraction, tall contact definitions must be the same for all corners.

Feedback

Figure 238 Top View of Tall Contact Layout



See Also

- CONDUCTOR
- VIA
- Table-Based Modeling of Tall Contacts

Feedback

CONTACT_WIDTH_AND_LENGTH

Specifies contact spacing values for tall contact modeling. Valid within a VIA block.

Syntax

```
CONTACT_WIDTH_AND_LENGTH {(wmin,lmin), (w2,l2), ... (wmax,lmax)}
```

Arguments

Argument	Description
(wmin,lmin), (w2,l2) ...	Typical combinations of width and length values, in ascending order, for table-based tall contact modeling. Maximum of 9 value pairs. Units: microns

Description

Tall contacts that are very large or that exhibit strong coupling capacitance to other tall contacts can be modeled by using a combination of options in the CONDUCTOR and VIA definitions in the ITF file. Figure 239 illustrates the tall contact parameters used in the definitions.

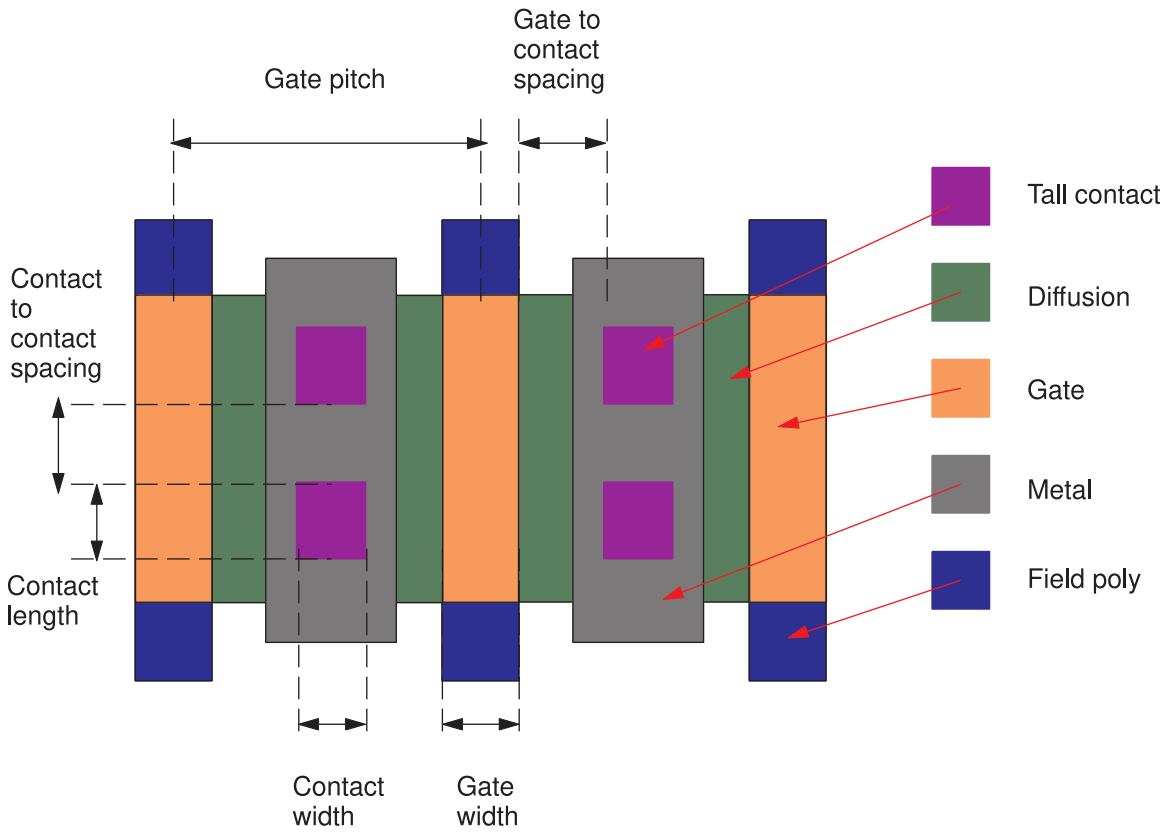
To model a tall contact with the table-based method, follow these steps:

- Include the following options in the VIA definition in the ITF file:
  - Set the LAYER_TYPE option to TALL_CONTACT.
  - (Optional) Set the DEVICE_TYPE option to a unique name. This name should be specified for one via layer and one gate conductor layer to indicate that they are used together in a tall contact structure.
  - (Optional) Include either or both of the CONTACT_WIDTH_AND_LENGTH and CONTACT_TO_CONTACT_SPACING options.
- Include the following options in the CONDUCTOR definition in the ITF file:
  - Set the LAYER_TYPE option to GATE.
  - (Optional) Set the DEVICE_TYPE option to the same name specified in the VIA definition.
  - (Optional) Include the GATE_PITCH, GATE_WIDTH, and GATE_TO_CONTACT_SPACING options.

For simultaneous multicorner (SMC) extraction, tall contact definitions must be the same for all corners.

Feedback

Figure 239 Top View of Tall Contact Layout



See Also

- CONDUCTOR
- VIA
- Table-Based Modeling of Tall Contacts

Feedback

CRT_VS_AREA

Specifies the temperature coefficients of resistance as a function of via area. Valid within a VIA block.

Syntax

```
CRT_VS_AREA {
    (area_1, crt1_1, crt2_2)
    (area_2, crt1_2, crt2_2)
    …
    (area_n, crt1_n, crt2_n)
}
```

Arguments

Argument	Description
area_1 … area_n	Via areas specified in increasing order Units: square microns
crt1_1 … crt1_n	Linear temperature coefficients for corresponding via sizes
crt2_1 … crt2_n	Quadratic temperature coefficients for corresponding via sizes

Description

Use the CRT_VS_AREA option within a VIA block to specify the temperature coefficients of resistance as function of via area. There is no limit to the number of entries. You cannot specify the CRT_VS_AREA option with either the CRT1 or CRT2 options in the same VIA block.

When the actual via size does not exactly equal any of the area entries in the CRT_VS_AREA table, CRT1 and CRT2 are determined by the following methods:

- If the actual via size is less than the smallest area entry in the CRT_VS_AREA table, the CRT values are set to the corresponding CRT1 and CRT2 entries of the smallest area entry; no extrapolation is performed.
- If the actual via size falls between two area entries in the CRT_VS_AREA table, CRT1 and CRT2 are calculated by linear interpolation.
- If the actual area is greater than the largest area entry in the CRT_VS_AREA table, the CRT values are set to the corresponding CRT1 and CRT2 entries of the largest area entry; no extrapolation is performed.

CRT and AREA values specified in a mapping file take precedence over the CRT_VS_AREA values specified in an ITF file.

Feedback

Errors

The grdgenxo tool issues a warning message if any of the following conditions is true:

- Area values in the table are not specified in increasing order. The tool internally reorders the table entries with increasing area values.
- The absolute value of CRT1 specified in the table is greater than 0.02.
- The value of CRT2 specified in the table is less than -0.002.

The grdgenxo tool issues an error message and stops the run if any of the following conditions is true:

- You specify both the `CRT_VS_AREA` option and the `CRT` option in the same `VIA` statement.
- You specify neither the nominal temperature for the via layer nor the global temperature.
- The `CRT_VS_AREA` table contains fewer than two rows. This same requirement applies to the `RPV_VS_AREA` table.
- The area values in the table are zero or negative.
- The `CRT_VS_AREA` option contains duplicate area values.
- You use the `-res_update` option with the `grdgenxo` command while adding or removing a `CRT_VS_AREA` table.

Examples

```
CRT_VS_AREA {
    (0.002025, 9.04E-04, 4.74E-07)
    (0.005265, 1.18E-03, 8.02E-07)
}
```

See Also

- CRT1, CRT2, and T0
- VIA

Feedback

CRT_VS_SI_WIDTH

Specifies CRT-based temperature derating for different conductor widths. Valid within a CONDUCTOR block.

Syntax

```
CRT_VS_SI_WIDTH {
      (siw_1, crt1_1, crt2_1)
      (siw_2, crt1_2, crt2_2)
      …
      (siw_n, crt1_n, crt2_n)
}
```

Arguments

Argument	Description
siw_1 … siw_n	Conductor silicon (post-etch) widths Units: microns
crt1_1 … crt1_n	Linear temperature coefficients for the corresponding conductor widths
crt2_1 … crt2_n	Quadratic temperature coefficients for the corresponding conductor widths

Description

Use a CRT_VS_SI_WIDTH table within a CONDUCTOR block to define CRT-based temperature derating for different conductor widths. There is no limit to the number of entries you can specify.

When the actual conductor width does not exactly equal any of the *siw* values in the CRT_VS_SI_WIDTH table, then CRT1 and CRT2 are determined by the following methods:

- If the actual conductor width is less than the smallest *siw* value in the CRT_VS_SI_WIDTH table, the CRT values are set to the corresponding *crt1* and *crt2* entries of the smallest *siw* entry; no extrapolation is performed.
- If the actual conductor width falls between two *siw* values in the CRT_VS_SI_WIDTH table, CRT1 and CRT2 are calculated by linear interpolation.
- If the actual conductor width is greater than the largest *siw* value in the CRT_VS_SI_WIDTH table, the CRT values are set to the corresponding *crt1* and *crt2* entries of the largest *siw* entry; no extrapolation is performed.

If both the CRT_VS_SI_WIDTH and RPSQ_VS_SI_WIDTH statements are specified for the same conductor, the width index should be the same for both statements.

Feedback

Examples

```
CONDUCTOR MET1 {
    THICKNESS=0.6 WMIN=0.34 SMIN=0.40
    CRT_VS_SI_WIDTH {
        (0.34,  0.001, 0.000)
        (0.40,  0.001, 0.001)
        (0.823, 0.002, 0.001)
        (2.0,   0.003, 0.001)
    }
}
```

See Also

- CONDUCTOR
- CRT1, CRT2, and T0
- RPSQ_VS_SI_WIDTH

Feedback

CRT1, CRT2, and T0

Defines parameters for temperature-dependent resistance models. Valid in `CONDUCTOR`, `VIA`, and `TSV` blocks.

Syntax

```
CRT1 = lin_coeff
CRT2 = quad_coeff
T0 = nominal_temp
```

Arguments

Argument	Description
lin_coeff	Linear temperature coefficient for the layer. Specified on a per-layer basis. Default: 0
quad_coeff	Quadratic temperature coefficient for the layer. Specified on a per-layer basis. Default: 0
nominal_temp	Nominal temperature for the layer Units: degrees Celsius Default: Temperature specified by the `GLOBAL_TEMPERATURE` keyword.

Description

The `CRT1`, `CRT2`, and `T0` options define temperature-dependent resistance models for conducting layers and vias. The resistances are modeled similar to the way they are modeled in SPICE, by using the following equation:

$$R = R0 \times \left[CRT1 \times (T\text{-}T0) + CRT2 \times (T\text{-}T0)^2 + 1 \right]$$

In this equation,

- R is the modeled resistance at the operating temperature T.
- R0 is the resistance value at the nominal temperature T0.
- CRT1 and CRT2 are the linear and quadratic temperature coefficients.

The modeled resistance R exactly equals the nominal resistance (R0) if T=T0 or if CRT1 and CRT2 both equal 0.

If either CRT1 or CRT2 is nonzero for a layer, a nominal temperature specification is required for that layer. Set a global value for nominal temperature with the

GLOBAL_TEMPERATURE option at the beginning of the ITF file. If you set a nominal temperature both globally and for an individual layer, the layer nominal temperature overrides the global setting.

The OPERATING_TEMPERATURE command must also be set in the StarRC command file to use the derating information in the nxtgrd file. If the resistance of a layer is changed by the mapping file, and if that layer has temperature derating in the ITF file, specifying the OPERATING_TEMPERATURE command uses the temperature derating coefficients for that layer from the ITF file.

Examples

```
TECHNOLOGY = example_tech
GLOBAL_TEMPERATURE = 31.0
DIELECTRIC IMD2 { THICKNESS=2.0 ER=3.9 }
CONDUCTOR metal2 {
   CRT1=3.00e-3 CRT2=2.0e-7
   THICKNESS = 0.6 SMIN=0.5 WMIN=0.5 RPSQ = 0.06
}
DIELECTRIC IMD1 { THICKNESS=1.9 ER=4.9 }
CONDUCTOR metal1 {
   CRT1=3.50e-3 CRT2=2.5e-7
   THICKNESS = 0.5 SMIN = 0.4 WMIN=0.4 RPSQ = 0.08
}
DIELECTRIC FOX { THICKNESS=1.0 ER=3.9 }
VIA via1 {
   FROM=metal1 TO=metal2 AREA=1 RPV=1
   CRT1=2.5e-3 CRT2=1e-6 T0=29
}
```

See Also

- GLOBAL_TEMPERATURE
- OPERATING_TEMPERATURE

Feedback

CUT_END_EXTENSION_TABLE

Specifies conductor line end extensions. Valid within a CONDUCTOR block.

Syntax

```
CUT_END_EXTENSION_TABLE
  PARALLEL_TO_REFERENCE | PERPENDICULAR_TO_REFERENCE
  {
  [NUMBER_OF_MASKS = num_masks]
  [MASKS(a,b) [(c,d)] {
    SPACINGS { s1 s2 … sn }
    VALUES   { v1 v2 ... vn }  }
  [MASKS(i,j) [(k,l)] { ... }]
  }
```

Arguments

Argument	Description
PARALLEL_TO_REFERENCE	Applies to wires that are parallel to the reference direction
PERPENDICULAR_TO_REFERENCE	Applies to wires that are perpendicular to the reference direction
num_masks	In a multiple mask patterning flow, the number of different mask colors
MASKS(*a*,*b*), (*c*,*d*), ...	In a multiple mask patterning flow, the pairs of mask numbers to which the extension table definition applies
s1, *s2*, ...	Spacing values specified in ascending order; the values must be the same for all tables Units: microns
v1, *v2*, ...	Extension values. The notation v(s1,w1) denotes the value corresponding to spacing *s1* and width *w1*. Units: microns

Description

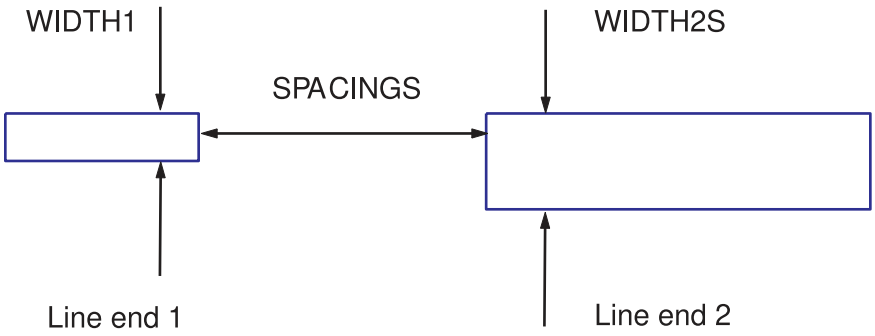
The CUT_END_EXTENSION_TABLE statement can only be used in conjunction with the LINE_END_EXTENSION_TABLE statement.

A line end extension table is a set of lookup tables that modifies drawn conductor dimensions before extraction. Figure 240 shows two line ends and the geometric parameters represented by the WIDTH1, WIDTH2S, and SPACINGS keywords of the LINE_END_EXTENSION_TABLE option. For each specified width of conductor line end 1, the

Feedback

table provides effective line extension values as a function of the distance to line end 2 and the width of line end 2.

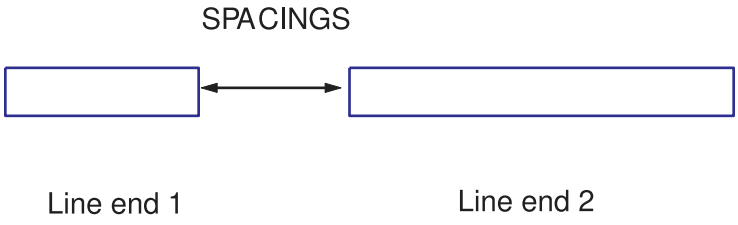
Figure 240 Line End Extension Parameters



You can optionally use the `CUT_END_EXTENSION_TABLE` statement to specify a one-dimensional table that provides line extension values as a function of the spacing between two lines, as illustrated in Figure 241. If present, the `CUT_END_EXTENSION_TABLE` statement is applied before the `LINE_END_EXTENSION_TABLE` statement.

The maximum spacing in the `CUT_END_EXTENSION_TABLE` statement is usually smaller than the spacings in the `LINE_END_EXTENSION_TABLE` statement.

Figure 241 Cut End Extension Parameters



Examples

The following example applies both cut end extensions and line end extensions for a process without multiple masks.

```
CONDUCTOR MetalX {
...
CUT_END_EXTENSION_TABLE PERPENDICULAR_TO_REFERENCE {
```

```
   SPACINGS {0.03}
   VALUES {0.002}   }
LINE_END_EXTENSION_TABLE PERPENDICULAR_TO_REFERENCE {
   NUMBER_OF_WIDTHS = 2
   WIDTH1 = 0.02 {
      SPACINGS {0.08 0.100 0.200}
      WIDTH2S {0.05 0.08 0.200}
      VALUES {0.00 0.00 0.00 0.00 0.00 0.001 0.002 0.003 0.004}
   }
   WIDTH1 = 0.04 {
      SPACINGS {0.08 0.100 0.200}
      WIDTH2S {0.05 0.08 0.200}
      VALUES {0.00 0.001 0.002 0.004 0.005 0.008 0.010 0.012 0.015}
   }
}
```

See Also

- LINE_END_EXTENSION_TABLE

Feedback

DAMAGE_ER

Defines the equivalent permittivity of a damage layer. Valid within a `DIELECTRIC` block.

Syntax

```
DAMAGE_ER = relative_permittivity
```

Arguments

Argument	Description
relative_permittivity	Relative permittivity

Description

Use the `DAMAGE_ER` to specify the relative permittivity of a dielectric damage layer. This option must be used with either the `SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING` or `DAMAGE_THICKNESS` option.

These options cannot be used for conformal dielectric layers.

See Also

- DAMAGE_THICKNESS
- SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING

Feedback

DAMAGE_THICKNESS

Specifies a fixed thickness of a the damage region of a planar dielectric layer. Valid within a `DIELECTRIC` block.

Syntax

```
DAMAGE_THICKNESS = thickness
```

Arguments

Argument	Description
thickness	Thickness of the damage layer on the surface of the dielectric Units: microns Range: 0.001 micron or greater

Description

Use the `DAMAGE_THICKNESS` option to specify a fixed thickness for the damage region of a planar dielectric layer.

To model the damage region thickness as a function of feature width and spacing, use the `SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING` option.

You must also use the `DAMAGE_ER` to specify the equivalent permittivity of the damage layer.

These options cannot be used for conformal dielectric layers.

See Also

- DAMAGE_ER
- SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING

Feedback

DENSITY_BOX_WEIGHTING_FACTOR

Specifies a density box weighting factor. Valid within a CONDUCTOR block.

Syntax

```
DENSITY_BOX_WEIGHTING_FACTOR {
      (s1 w1) [(s2 w2) ... [(s5 w5)]]
}
```

Arguments

Argument	Description
s1 ... s5	The size of the density box, a square with sides of length s. Values s1 through s5 are positive values, listed in ascending order. Maximum of 5 boxes allowed. Maximum value: 500 Default: One box with size 50 Units: microns
w1 ... w5	The weighting factor; a floating-point number. If W is 0, that box is ignored. Range: -10 to +10 Default: 1 Units: none

Description

The DENSITY_BOX_WEIGHTING_FACTOR option defines regions for evaluating conductor feature densities for use with several options that model conductor thickness variation.

Using Density Boxes With the POLYNOMIAL_BASED_THICKNESS_VARIATION and THICKNESS_VS_DENSITY_AND_WIDTH Options

The POLYNOMIAL_BASED_THICKNESS_VARIATION option models conductor thickness variation with an equation-based approach that includes feature density.

The THICKNESS_VS_DENSITY_AND_WIDTH option models conductor thickness variation with respect to feature density and feature width using a two-dimensional lookup table.

By default, feature density is calculated within a square box 50 microns on a side, centered on the conductor of interest. You can change the size of this box by using the DENSITY_BOX_WEIGHTING_FACTOR option. You must specify only one density box and set the weighting factor to 1.

Using Density Boxes With the THICKNESS_VS_DENSITY Option

The THICKNESS_VS_DENSITY option allows you to model conductor thickness as a function of feature density in up to five zones around the conductor. This approach might be

suitable for complex thickness variation patterns such as those created by the chemical-mechanical polishing process.

Calculate conductor thickness as a function of the feature density as follows:

- Use the DENSITY_BOX_WEIGHTING_FACTOR option to specify the size and weighting factors for up to five density boxes (evaluation regions), which are square boxes centered around the conductor of interest.

  If you do not specify the DENSITY_BOX_WEIGHTING_FACTOR option, the default is a single density box with a size of 50 µm and a weighting factor of 1.

- Use the THICKNESS_VS_DENSITY option to specify density values and the thickness variations that correspond to those densities.

Analysis is performed as follows:

1. Based on the layout, determine the conductor feature density within a density box
2. Scale the density by the density box weighting factor
3. Repeat steps 1 and 2 for each supplied density box
4. Determine the effective density by adding the scaled density values
5. Look up the thickness variation for the effective density
6. Multiply the thickness variation to the nominal thickness to calculate the adjusted thickness
7. Calculate resistance and capacitance based on the adjusted conductor thickness

Examples

```
CONDUCTOR metal3 {
     SMIN= 0.35 WMIN=0.42 THICKNESS=0.53 RPSQ=0.061
     THICKNESS_VS_DENSITY {(0.1 -0.1)(0.2 0.1)(0.3 0.15)(0.5 0.3)}
     DENSITY_BOX_WEIGHTING_FACTOR
         { (10 1)(20 0.23)(30 0.29)(40 0.08)(50 0.12) }
}
```

See Also

- THICKNESS_VS_DENSITY
- THICKNESS_VS_DENSITY_AND_WIDTH
- POLYNOMIAL_BASED_THICKNESS_VARIATION

Feedback

DEVICE_TYPE

Identifies layers that are associated with a specific device type. Valid within a CONDUCTOR or VIA definition.

Syntax

```
DEVICE_TYPE { [device_type_name_1] … [device_type_name_2] | ALL | NONE }
```

Arguments

Argument	Description
device_type_name	Applies the layer properties to the specified device type. Valid within a CONDUCTOR or VIA block.
ALL	Applies the layer properties to all device types Valid only within a CONDUCTOR block.
NONE	Does not apply the layer properties to any device types. Useful for conductor layers used for capacitors or resistors that are never in or near a device. Valid only within a CONDUCTOR block.

Description

The DEVICE_TYPE option indicates that conductor or via layers are associated with a specific device type.

Via Layers

For use with via layers, the DEVICE_TYPE option indicates that the via layer is a tall contact layer. A tall contact requires one via layer and one gate conductor layer. For both layers, the DEVICE_TYPE option must be set to the same name. The conductor layer must include a LAYER_TYPE=GATE specification.

Conductor Layers

For use with conductor layers, the DEVICE_TYPE option has the following constraints:

- The DEVICE_TYPE designation can only be included in conductors with GATE, FIELD_POLY, TRENCH_CONTACT, or DIFFUSION layer types.
- Only one DEVICE_TYPE option is allowed for each conductor.
- Exactly one conductor with a LAYER_TYPE=GATE statement and one conductor with a LAYER_TYPE=DIFFUSION statement must be specified for each defined device type name.

Feedback

- Conductors that have no specified device type but have a LAYER_TYPE=GATE statement are applied to all device types, therefore no other conductors with a LAYER_TYPE=GATE statement are allowed to exist in the process if a DEVICE_TYPE statement appears anywhere in the ITF file. An equivalent rule is applied to conductors that have no device type but do have a LAYER_TYPE=DIFFUSION statement.
- Any number of conductors that have a LAYER_TYPE=FIELD_POLY or LAYER_TYPE=TRENCH_CONTACT statement can be associated with a single device type name.
- DEVICE_TYPE NONE is valid only for conductors with LAYER_TYPE=FIELD_POLY or LAYER_TYPE=TRENCH_CONTACT statements.

Examples

The following example shows part of an ITF file that uses the DEVICE_TYPE option.

Example 40 ITF File With DEVICE_TYPE Option

```
CONDUCTOR TC_RSD { DEVICE_TYPE { N_RSD P_RSD } … }
CONDUCTOR TC { DEVICE_TYPE { N_NO_RSD P_NO_RSD } … }
…
CONDUCTOR FPOLY_N { DEVICE_TYPE { N_RSD N_NO_RSD } … }
CONDUCTOR FPOLY_P { DEVICE_TYPE { P_RSD P_NO_RSD } … }
CONDUCTOR GPOLY_N { DEVICE_TYPE { N_RSD N_NO_RSD } … }
CONDUCTOR GPOLY_P { DEVICE_TYPE { P_RSD P_NO_RSD } … }
…
CONDUCTOR DIFF_NO_RSD { DEVICE_TYPE { N_NO_RSD P_NO_RSD } … }
CONDUCTOR DIFF_N_RSD { DEVICE_TYPE { N_RSD } … }
CONDUCTOR DIFF_P_RSD { DEVICE_TYPE { P_RSD } … }
```

See Also

- LAYER_TYPE
- CONDUCTOR
- VIA

Feedback

DIELECTRIC

Describes the properties of a dielectric layer.

Syntax

```
DIELECTRIC dielectric_name {
   ER = er_value
   [ER_VS_SI_SPACING  {...}]
   [ER_TABLE { … }]
   [THICKNESS = diel_thickness]    |
      [MEASURED_FROM = meas_layer | TOP_OF_CHIP
         [SW_T = sw_thick] [TW_T = tw_thick] ]   |
      [IS_CONFORMAL
         [ASSOCIATED_CONDUCTOR = conductor_name
            [BW_T = bw_thick] [SPACER_ER_VS_WIDTH_AND_SPACING {...}]
         ]
         [SW_T = sw_thick | SW_T_VS_WIDTH_AND_SPACING {...}]
         [TW_T = tw_thick]
      ]
   ]
 [THICKNESS_VS_SPACING  {...}]
   [DAMAGE_ER = damg_er DAMAGE_THICKNESS = damg_thk
  | DAMAGE_ER = damg_er SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING {...}]
}
```

Arguments

Argument	Description
dielectric_name	Name of the dielectric layer
er_value	Relative permittivity of the layer
diel_thickness	Thickness of the layer Units: microns
meas_layer	Name of the reference dielectric layer
sw_thick	Sidewall thickness Units: microns
tw_thick	Top wall thickness Units: microns
bw_thick	Bottom wall thickness Units: microns
conductor_name	Name of the associated conductor

Feedback

Argument	Description
damg_thk	Damage layer thickness Units: microns
damg_er	Relative permittivity of the damage layer

Description

The DIELECTRIC statement describes a dielectric layer above or below a conductor layer.

Errors

The following error messages are issued when limitations for THICKNESS, TW_T, and SW_T are not observed:

```
ERROR:(908) ITF**
ERROR: Too thin SW_T value of 0.001 is specified for layer local1;
0 < SW_T < 0.005 is not allowed

ERROR:(910) ITF**
ERROR: Too thin TW_T value of 0.001 is specified for layer local1;
0 < TW_T < 0.005 is not allowed

ERROR:(906) ITF**
Too thin THICKNESS value of 0.0007 is specified for layer thin;
0<THICKNESS<0.001 is not allowed (THICKNESS=0 is allowed for conformal
dielectrics)
```

Examples

The following example describes a dielectric layer with a thickness of 0.3 µm and a relative permittivity of 3.9.

```
DIELECTRIC FOX { THICKNESS=0.3 ER=3.9 }
```

Feedback

DIELECTRIC_FILL_EMULATION_VS_SI_SPACING

Specifies dielectric fill for use in TLUPlus files. Valid within a CONDUCTOR block for routing layers only.

Syntax

```
DIELECTRIC_FILL_EMULATION_VS_SI_SPACING {
    SI_SPACINGS { s1 s2 … sm }
    LENGTHS  { L1 L2 … Ln }
    VALUES  { er(s1,L1) er(s2,L1) … er(sm,L1)
              er(s1,L2) er(s2,L2) … er(sm,L2)
              …
              er(s1,Ln) er(s2,Ln) … er(sm,Ln)
    }
}
```

Arguments

Argument	Description
s1 s2 … sm	Conductor spacings specified in ascending order Units: microns
L1 L2 … Ln	Conductor lengths specified in ascending order Units: microns
er(s1,L1) ... er(sm,Ln)	Relative permittivity for corresponding spacing and length values Units: none

Description

The DIELECTRIC_FILL_EMULATION_VS_SI_SPACING option provides a method for estimating the parasitic capacitance of routing layers for processes that use dielectric fill, which is the insertion of low-permittivity dielectric shapes to reduce capacitance.

Caution:

TLUPlus models are used by the parasitic extractor in other Synopsys tools, including the IC Compiler, IC Compiler II, Fusion Compiler, and Design Compiler Topographical Mode tools. Fill emulation is intended to be used early in the design cycle. Fill emulation is not intended for use in a signoff flow because emulated fill does not represent the actual design.

If the DIELECTRIC_FILL_EMULATION_VS_SI_SPACING option is present in an ITF file or in an nxtgrd file, the StarRC tool ignores it during extraction.

Feedback

Use this option as follows:

1. Insert the DIELECTRIC_FILL_EMULATION_VS_SI_SPACING option into the ITF file, for conductor routing layers only.
2. Use the grdgenxo tool to create a TLUPlus file from the ITF file.
3. Use the TLUPlus file in a place-and-route or synthesis tool.

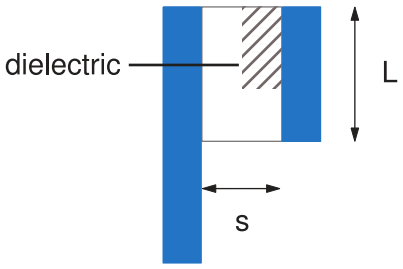
This option cannot be used with the following options in the same CONDUCTOR layer:

- LATERAL_CAP_SCALING_VS_SPACING
- ER_VS_SI_SPACING

The conductor spacing and length values are indexes for the two-dimensional table of relative permittivity values. Figure 242 shows the conductor spacing and length values required for the LENGTHS and SPACINGS lists. If the conductors have different lengths, the length of the shortest conductor is used for the table lookup.

If the actual spacing or length value does not match any of the values supplied in the table, but is within the specified range, linear interpolation is used. If the actual spacing or length value falls outside of the specified range, the nearest value in the table is used.

Figure 242 Application of Emulated Dielectric Fill



Examples

In the following example, when the spacing is 0.1 microns and the length is 5 microns, the relative permittivity is 3.0. For spacing of 0.2 microns and length of 10 microns, the relative permittivity is 2.7. If the length is 30 microns, the values for a length of 20 microns are used.

```
DIELECTRIC_FILL_EMULATION_VS_SI_SPACING {
    SPACINGS { 0.1 0.2 }
    LENGTHS  { 5.0 10.0 20.0 }
    VALUES   { 3.0 2.5 2.0 2.8 2.7 2.5}
}
```

See Also

- LATERAL_CAP_SCALING_VS_SI_SPACING

Feedback

DIELECTRIC_FILL_VS_SI_SPACING

Defines parameters of dielectric fill features. Valid within a CONDUCTOR block.

Syntax

```
DIELECTRIC_FILL_VS_SI_SPACING {
    SPACINGS { s1 s2 .. sn }
    WIDTHS { w(s1) w(s2) … w(sn) }
    THICKNESSES { t(s1) t(s2) … t(sn) }
    BOTTOM_HEIGHTS { h(s1) h(s2) … h(sn) }
    SIDE_TANGENTS { a(s1) a(s2) … a(sn) }
    ERS { e(s1) e(s2) … e(sn) }
}
```

Arguments

Argument	Description
s1 s2 … sn	Spacing values between the two conductors, in ascending order Units: microns Range: SMIN value to 5 times the SMIN value
w(s1) w(s2) … w(sn)	Width of the dielectric fill region Units: microns
t(s1) t(s2) … t(sn)	Thickness of the dielectric fill region Units: microns
h(s1) h(s2) … h(sn)	Bottom height: offset of the bottom of the dielectric fill region with respect to the drawn bottom height of the conductor. A negative value indicates that the dielectric fill region is lower than the conductor bottom height Units: microns
a(s1) a(s2) … a(sn)	Side tangent of the dielectric fill (defining the sidewall angle) Units: none
e(s1) e(s2) … e(sn)	Relative dielectric constant of the dielectric fill region Units: none

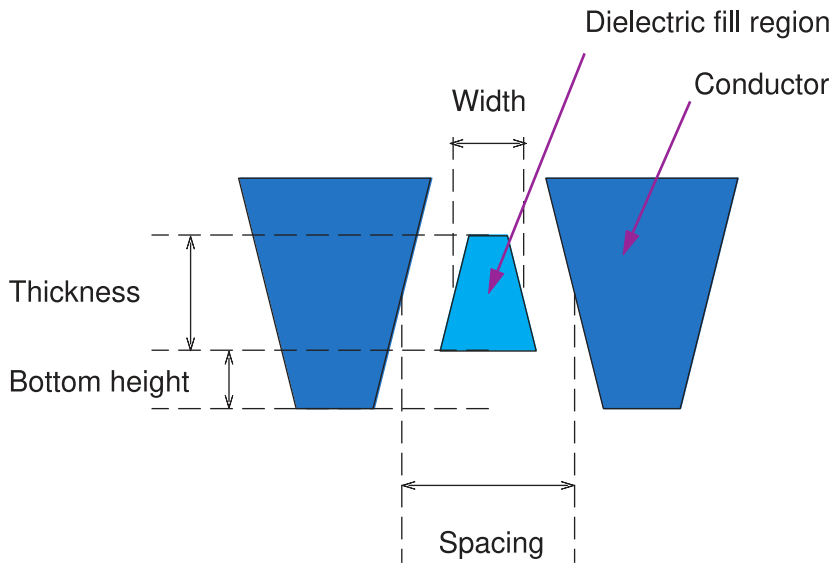
Description

The DIELECTRIC_FILL_VS_SI_SPACING command defines the parameters of dielectric fill. Dielectric fill features typically have very low dielectric constants and are inserted to reduce the coupling capacitance between upper-layer conductors. Figure 243 shows the geometry of a dielectric fill region.

The DIELECTRIC_FILL_GDS_FILE and DIELECTRIC_FILL_GDS_LAYER_MAP_FILE commands in the StarRC command file define the dielectric fill shapes referenced by the DIELECTRIC_FILL_VS_SI_SPACING command in the ITF file. These three commands must be used together.

The side tangent value represents the angular shift from vertical of the side of the dielectric fill feature. The use of side tangent values is similar to the SIDE_TANGENT option used at the top level of the CONDUCTOR definition. A positive side tangent results in a top width that is larger than the center width, while a negative value results in a top width that is smaller than the center width. The feature in Figure 243 results from a negative side tangent value.

Figure 243 Dielectric Fill Geometry and Parameters



Dielectric fill is inserted whenever the spacing between conductor features falls within the range specified in the SPACINGS list.

The geometry of a specific dielectric fill feature is determined as follows:

- If the actual conductor spacing is smaller than the smallest value in the SPACINGS list, no dielectric fill is inserted.
- If the actual conductor spacing is larger than the largest value in the SPACINGS list, dielectric fill is inserted using the parameters for the largest spacing value in the SPACINGS list.

- If the actual conductor spacing falls between two spacing values in the SPACINGS list, linear interpolation is applied to calculate the width, thickness, and bottom height.
- If the actual conductor spacing falls between two spacing values in the SPACINGS list, the dielectric constant is not interpolated, but instead is set equal to the ERS value that corresponds to the smaller of the two spacing values.

For each conductor spacing value in the SPACINGS list, you must provide one value of each of the other parameters. In other words, the number of entries in each of the parameter lists must be equal. Lists of values are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Examples

In the following example, dielectric fill is inserted between features in conductor layer M1 whenever the spacing between conductor features is greater than 0.035 microns.

```
CONDUCTOR M1 {
THICKNESS=0.08
CRT1=1E-03 CRT2=2E-07
SIDE_TANGENT=0.09
POLYNOMIAL_BASED_THICKNESS_VARIATION { ... }
ETCH_VS_WIDTH_AND_SPACING { ...}
DIELECTRIC_FILL_VS_SI_SPACING {
     SPACINGS { 0.035 .07 .24 }
     WIDTHS { 0.03 0.06 0.12 }
     THICKNESSES    { 0.22 0.44 0.88 }
     BOTTOM_HEIGHTS { 0.008 0.016 0.032 }
     SIDE_TANGENTS { 0.09 0.095 0.15 }
     ERS { 2.6 2.7 2.8 }
}
```

See Also

- CONDUCTOR
- SIDE_TANGENT
- DIELECTRIC_FILL_GDS_FILE
- DIELECTRIC_FILL_GDS_LAYER_MAP_FILE

Feedback

DROP_FACTOR

Specifies the decrease in base height of all upper conductors when the bottom conductor is not present in the given layout area. Valid within a CONDUCTOR block.

Syntax

```
DROP_FACTOR = drop_value
```

Arguments

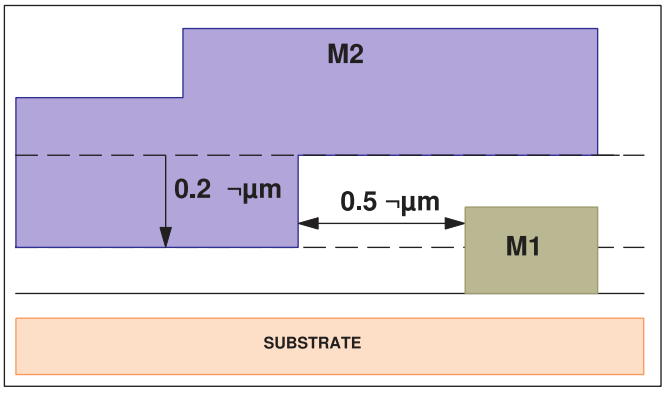
Argument	Description
drop_value	Conductor base height decrease due to missing lower-level conductor Units: microns

Description

The DROP_FACTOR option specifies the decrease in base height of upper-level conductors when a lower-level conductor is not present. Use the DROP_FACTOR option in the CONDUCTOR block for the lower-level conductor. You can specify a drop factor for no more than four conductors. You cannot use drop factors with simultaneous multicorner extraction.

Figure 244 shows the effect of setting the drop factor to 0.2 microns for lower-level conductor M1. The base height of upper-level conductor M2 drops by 0.2 microns when M1 is not present. A lateral gap is maintained between the dropped part of the upper-level conductor and the lower-level conductor. This lateral gap is 0.5 um by default, but can be modified by using the DROP_FACTOR_LATERAL_SPACING statement.

Figure 244 Drop Factor Example



Feedback

See Also

- DROP_FACTOR_LATERAL_SPACING

Feedback

DROP_FACTOR_LATERAL_SPACING

Specifies a constant lateral spacing value to use in conjunction with conductor drop factors. Valid within a `CONDUCTOR` block.

Syntax

```
DROP_FACTOR_LATERAL_SPACING = drop_lateral
```

Arguments

Argument	Description
drop_lateral	The lateral spacing; 0.5 to 4.0 Units: microns Default: 0.5

Description

The `DROP_FACTOR_LATERAL_SPACING` statement specifies a constant lateral spacing value used between a lower-level conductor that has a `DROP_FACTOR` statement and the dropped part of an upper-level conductor.

Specify the `DROP_FACTOR_LATERAL_SPACING` statement in the global parameters section after the `TECHNOLOGY` statement.

Examples

This example specifies a lateral spacing of 0.6 µm for all conductors when dropped.

```
TECHNOLOGY = example_tech
DROP_FACTOR_LATERAL_SPACING = 0.6
```

See Also

- DROP_FACTOR

Feedback

DUAL_POLY

Specifies the name of a polysilicon layer associated with the current layer. Valid only within a `CONDUCTOR` block.

Syntax

```
DUAL_POLY = top_poly_namd
```

Arguments

Argument	Description
top_poly_name	For a dual polysilicon gate process, the name of the top polysilicon layer. This option is valid only when defining a bottom polysilicon layer with `LAYER_TYPE=GATE`.

Description

Some processes use dual polysilicon layers for the transistor gate conductor layer, in conjunction with a tall contact process.

Model a dual polysilicon gate in the ITF file as follows:

- Define the properties of the top polysilicon layer as an ITF conductor layer. Do not include the `LAYER_TYPE` option for this layer.
- Define the properties of the bottom polysilicon layer as another ITF conductor layer and set the `LAYER_TYPE` option to `GATE`. Set the `DUAL_POLY` option to the name of the top polysilicon layer.

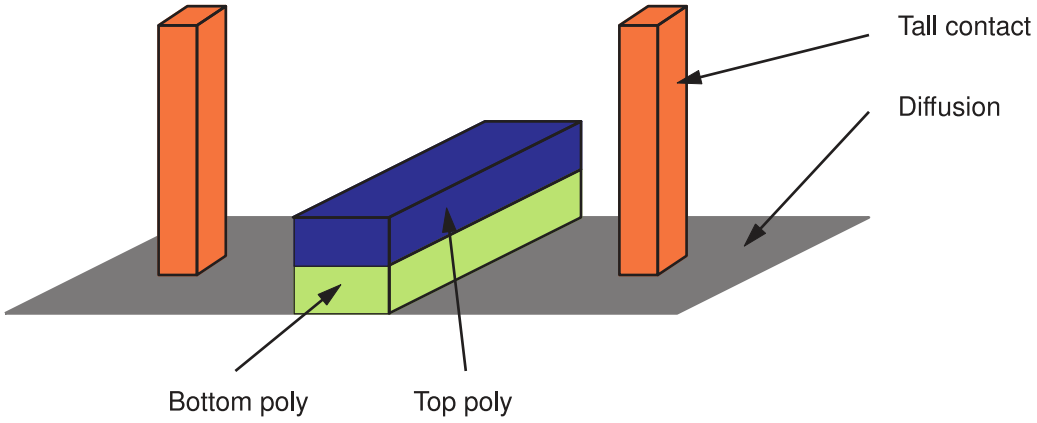
This process is only valid when used with tall contacts. The contact height must be at least ten times the combined thickness of the top and bottom polysilicon layers.

The top and bottom polysilicon layers must meet the following conditions:

- The layers must be separated by a vertical distance of less than 15 Angstroms.
- The layers must have the same etch value or the same set of etch tables in their ITF file definitions.
- The layers must share the same ignore capacitance properties in the `ignore_cap_layers` section of the mapping file.
- The layout polygons must have the same width (the dimension along the transistor channel length direction).

Feedback

Figure 245 Dual Polysilicon Gate Transistor With Tall Contacts



See Also

- CONDUCTOR
- Dual Polysilicon Gate Process

Feedback

ER

Specifies the relative permittivity of a dielectric. Valid within a `DIELECTRIC` block.

Syntax

```
ER = permittivity
```

Arguments

Argument	Description
permittivity	Relative permittivity of a dielectric layer

Description

The `ER` parameter specifies the relative permittivity, or dielectric constant, of a dielectric layer.

Examples

```
DIELECTRIC D2 {THICKNESS = 1.2 ER = 3.9}
```

See Also

- DIELECTRIC

Feedback

ER_TABLE

Specifies the device-dependent relative permittivity of a dielectric.

Syntax

```
ER_TABLE {
  (device_type1 permittivity1)
  (device_type2 permittivity2)
  …
}
```

Arguments

Argument	Description
device_type	Device type
permittivity	Relative permittivity of a dielectric layer

Description

The ER_TABLE option specifies the device-dependent relative permittivity, or dielectric constant, of a dielectric layer.

Specify the ER_TABLE option within the DIELECTRIC statement for the gate oxide under the polysilicon layer.

In the mapping file, you must provide the mapping information for the different device types.

Examples

The following example shows the use of the ER_TABLE option:

```
DIELECTRIC D3_BOT1 {
  THICKNESS=0.0 IS_CONFORMAL
  ER=7.55 ASSOCIATED_CONDUCTOR=POLY
  SW_T=0 TW_T=0 BW_T=0.001
  ER_TABLE { (G_1D5VIO_PMOS 7.0) (G_CORE_PMOS 8.0) }
}
```

See Also

- DIELECTRIC
- ER

Feedback

ER_VS_SI_SPACING

Specifies dielectric relative permittivity (ER) values as a function of conductor spacing width. Valid within a `DIELECTRIC` block.

Syntax

```
ER_VS_SI_SPACING {
  ( spacing_1, er_1 )
  ( spacing_2, er_2 )
  …
  ( spacing_n, er_n )
}
```

Arguments

Argument	Description
spacing_1 ... *spacing_n*	Spacing values specified in ascending order Units: microns
er_1 ... *er_n*	Relative permittivity for the corresponding spacing values Units: none

Description

The `ER_VS_SI_SPACING` option models variable dielectric constants in the ITF file. One application is for modeling of capacitance variation due to double patterning processes. If two conductors move closer together or farther apart, the change in capacitance can be modeled by varying the dielectric constant.

To enable double patterning extraction, set the `DPT` command in the StarRC command file to `YES` and specify the `ER_VS_SI_SPACING` option within a `DIELECTRIC` block in the ITF file.

An ER value is valid for all spacings between the spacing listed in the same value pair and the spacing listed in the previous value pair. The first ER value is valid for all spacings up to and including the first spacing value.

Examples

A dielectric layer might have the following relative permittivity definition:

```
ER = 5.3
ER_VS_SI_SPACING {
  ( 0.020, 7.2 )
  ( 0.050, 6.4 )
  ( 0.070, 6.0 )
}
```

The resulting permittivity values are shown in Table 98.

Table 98 Relative Permittivity for Different Spacing Values

Spacing Value S	Relative Permittivity
0 < S <= 0.02	7.2
0.02 < S <= 0.05	6.4
0.05 < S <= 0.07	6.0
> 0.07	5.3

See Also

- DIELECTRIC
- DPT
- ER

Feedback

ETCH

Specifies a width adjustment to model layer etch effects. Valid within a CONDUCTOR block.

Syntax

```
ETCH = etch_value
```

Arguments

Argument	Description
etch_value	Absolute width adjustment for one sidewall. A positive value shrinks the conductor; a negative value expands it. Units: microns

Description

The ETCH option applies an etch value to each of the sidewalls of a conductor. A positive argument denotes conductor shrink; a negative value denotes conductor expansion. The adjusted conductor width is equal to the drawn width minus twice the etch value. The new conductor width is determined before resistance is calculated (for example by applying the RPSQ keyword).

You can specify that an etch operation is to be used for only capacitance calculations by using the CAPACITIVE_ONLY_ETCH option instead of the ETCH option. Similarly, you can specify that an etch operation is to be used for only resistance calculations by using the RESISTIVE_ONLY_ETCH option instead of the ETCH option.

If you use one of the ETCH options in addition to one or more ETCH_VS_WIDTH_AND_SPACING tables, the ETCH_VS_WIDTH_AND_SPACING operations are applied first, followed by the ETCH operation.

Examples

```
CONDUCTOR M1 {
     THICKNESS=0.6 WMIN=0.3 SMIN=0.3 RPSQ=0.05 ETCH=0.05
}
```

See Also

- CAPACITIVE_ONLY_ETCH
- RESISTIVE_ONLY_ETCH
- ETCH_VS_WIDTH_AND_SPACING

Feedback

ETCH_VS_CONTACT_AND_GATE_SPACINGS

Specifies via etch as a function of spacing between vias and spacing between the gate and via. Valid within a `VIA` block.

Syntax

```
ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {
     [NUMBER_OF_TABLES = num_of_tables]
     name_of_table1 {
     CONTACT_TO_CONTACT_SPACINGS { c1 c2 ... cn }
     GATE_TO_CONTACT_SPACINGS    { s1 s2 ... sm }
     VALUES { v(c1,s1) v(c2,s1) ... v(cn,s1)
              v(c1,s2) v(c2,s2) ... v(cn,s2)
              …
              v(c1,sm) v(c2,sm) ... v(cn,sm)   }
    [name_of_table2 {
     CONTACT_TO_CONTACT_SPACINGS { c1 c2 ... cn }
     GATE_TO_CONTACT_SPACINGS    { s1 s2 ... sm }
     VALUES { v(c1,s1) v(c2,s1) ... v(cn,s1)
              v(c1,s2) v(c2,s2) ... v(cn,s2)
              …
              v(c1,sm) v(c2,sm) ... v(cn,sm)   }   ]
     …
}
```

Arguments

Argument	Description
`CAPACITIVE_ONLY`	Applies etch effect to capacitance only
num_of_tables	The number of supplied tables (default is 1)
nam_of_tableX	The name of the subsequent table; allowed only if multiple tables are specified
c1 c2 c3 ...	Contact to contact spacing values specified in ascending order Units: microns
s1 s2 s3 ...	Gate to contact spacing values specified in ascending order Units: microns
v(c1,s1) v(c2,s2) ...	Etch values of corresponding contact and gate spacing pairs Units: microns

Feedback

Description

The actual size of contacts in silicon might be different than the sizes drawn in layout. To account for this difference during parasitic extraction, the `VIA` statement allows a contact bias (etch) to be specified as a function of contact-to-contact and gate-to-contact spacing. The etch affects only the estimated capacitance.

A positive etch value models a shrinking contact; a negative etch value models an expanding contact.

Errors

Common error and warning conditions are as follows:

- From the grdgenxo tool
  - If the `NUMBER_OF_TABLES` keyword is not specified, an error occurs if there are multiple tables or if a table has a table name.
  - If the number of tables is different from the number specified with the `NUMBER_OF_TABLES` keyword, the grdgenxo tool stops and issues an error message.
  - If the old ITF format (no `NUMBER_OF_TABLES`, no table name) and the new format exist at the same time in the ITF file, with each format used for contact-etch and Cf tables, the grdgenxo tool stops and issues an error message.
- From the StarRC tool
  - If the `table_name` option in a mapping file does not match any table name in the ITF file, the StarRC tool issues a warning message.
  - If the `table_name` option is used in the mapping file for a layer other than a gate layer, the StarRC tool issues a warning message.
  - If contact-etch or gate-diffusion capacitance tables with a table name are present in the ITF file, but a gate-level layer in the mapping file does not have the `table_name` option, the StarRC tool issues a warning message.

Examples

The following `VIA` definition does not have a contact bias:

```
VIA DiffCont {
     FROM=diff TO=metal1 AREA=0.003 RPV=15
     CRT1=6.56e-04 CRT2=-5.643e-0
}
```

The via definition can be extended to include a contact bias 2-D table, or etch table, as a function of contact-to-contact and gate-to-contact spacings. For example,

```
VIA DiffCont {
     FROM=diff TO=metal1 AREA=0.003 RPV=15
```

```
      CRT1=6.56e-04 CRT2=-5.643e-0
      ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {
           CONTACT_TO_CONTACT_SPACINGS {0.1 0.2 0.3}
           GATE_TO_CONTACT_SPACINGS {0.05 0.1 0.15}
           VALUES {0.005 0.006 0.007
                   0.004 0.005 0.006
                   0.003 0.004 0.005}
      }
}
```

You can specify multiple table definitions in the `ETCH_VS_CONTACT_AND_GATE_SPACINGS` option. Each table should have a name that associates with a gate database layer through a mapping file. You must use the same name for a contact etch table and a gate diffusion table for each kind of device. A keyword can, however, associate with only one contact-etch table or gate-diffusion capacitance table.

The following is an example of an ITF file with multiple contact etch tables defined:

```
VIA diffCont {
      FROM=diff TO=metal1 AREA=0.0036 RPV=55
      CRT1=9.56e-04 CRT2=-3.68e-07
      ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {
           NUMBER_OF_TABLES=2
           NMOS {
                CONTACT_TO_CONTACT_SPACINGS {0.08 0.12}
                GATE_TO_CONTACT_SPACINGS {0.04 0.08}
                VALUES {0.008 0.009 0.003 0.005}
           }
           PMOS {
                CONTACT_TO_CONTACT_SPACINGS {0.08 0.12}
                GATE_TO_CONTACT_SPACINGS {0.04}
                VALUES {0.004 0.002}
           }
       }
 }
```

Mapping File Syntax

Use the following syntax for the mapping file:

```
conducting_layers
        gate_database_layer gate_grd_layer [table_name=keyword]
```

If the `table_name` keyword is defined for a gate database layer, the StarRC tool finds and uses the contact etch table and gate-diffusion capacitance table of the same name in the ITF file.

If the `table_name` keyword is not specified for a gate database layer, no contact etch or gate-diffusion capacitance calculation is applied to the device with the gate of that data layer. For those devices, the contact etch is zero, and the gate-diffusion capacitance is

Feedback

the extracted value if the IGNORE_CAPACITANCE command is set to extract gate-diffusion coupling.

Mapping File for Multiple Contact Etch Table Support

When multiple contact etch tables are defined in the ITF file, the device gate layer that maps to the corresponding table name must be specified in the mapping file. Use the following mapping file syntax to look up the appropriate gate-to-diffusion table:

```
conducting_layers
    gate_database_layer1 gate_grd_layer1 [table_name=name1]
    gate_database_layer2 gate_grd_layer2 [table_name=name2]
```

If table_name is defined for a gate, the StarRC tool finds and uses the corresponding gate-diffusion capacitance table with same name in the ITF file. If table_name is not defined for a gate database layer, no gate-diffusion capacitance calculation is done for the device with the gate of that database layer.

The following example shows a mapping file to look up the corresponding gate-to-diffusion capacitance tables based on the tables specified in the previous example:

```
conducting_layers
    ngate    gpoly   table_name=NMOS
    pgate    gpoly   table_name=PMOS
    tngate   gpoly
    tpgate   gpoly
```

With this mapping file definition, devices with ngate as the gate polysilicon use the NMOS contact etch table, and devices with pgate as the gate polysilicon use the PMOS table in the ITF file. No gate-to-contact etch is applied to the devices with tngate or tpgate gates.

See Also

- CAPACITIVE_ONLY_ETCH
- GATE_TO_DIFFUSION_CAP

Feedback

ETCH_VS_WIDTH_AND_LENGTH

Specifies different via etch values for the length and width sides of nonsquare vias. Valid within a VIA block.

Syntax

```
ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY {
    LENGTHS { L1 L2 … Ln }
    WIDTHS  { w1 w2 … wm }
    VALUES  { (eL_L1_w1, ew_L1_w1) (eL_L2_w1, ew_L2_w1) …
                                                … (eL_Ln_w1, ew_Ln_w1)
              (eL_L1_w2, ew_L1_w2) (eL_L2_w2, ew_L2_w2) …
                                                … (eL_Ln_w2, ew_Ln_w2)
              …
              (eL_L1_wm, ew_L1_wm) (eL_L2_wm, ew_L2_wm) …
                                                … (eL_Ln_wm, ew_Ln_wm)
    }
}
```

Arguments

Argument	Description
L1 L2 …	Via lengths specified in ascending order Units: microns
w1 w2 …	Via widths specified in ascending order Units: microns
(eL_L1_w1, ew_L1_w1) ...	A pair of etch values associated with the corresponding length and width indexes. The first value is the length etch (the etch value to be applied to each of the long sides). The second value is the width etch (the etch value to be applied to each of the short sides). The two etch values must be enclosed in parentheses and separated by a space or comma. Units: microns Range: 0 to one half of the corresponding dimension

Description

Specify the ETCH_VS_WIDTH_AND_LENGTH option within a VIA statement to apply different via etch values to the length and width sides of nonsquare vias.

For this command, the via length is the larger layout dimension and the via width is the smaller layout dimension, regardless of orientation.

The length and width values specified in the LENGTHS and WIDTHS keyword arguments are used as indexes for the table of etch values. Each combination of length *L* and width *w* has

a corresponding pair of etch values *(eL_L_w, ew_L_w)*, where *eL* is the length etch value and *ew* is the width etch value.

For example, consider a via with length 0.2 microns and width 0.15 microns. The table entry for this combination of dimensions is (0.005, 0.002). The post-etch via length is 0.19 microns, calculated by subtracting twice the length etch from the layout length (0.200 - 0.005 - 0.005). Similarly, the post-etch via width is 0.146 microns (0.15 - 0.002 - 0.002).

The following restrictions apply when using an `ETCH_VS_WIDTH_AND_LENGTH` table:

- When the length and width values specified in the `LENGTHS` and `WIDTHS` keyword arguments result in a combination for which the width is greater than the length, the StarRC tool sets the corresponding etch values to zero and issues a warning message.
- When the length and width values specified in the `LENGTHS` and `WIDTHS` keyword arguments are the same, the corresponding length etch and width etch values should be the same. If they are not the same, the tool issues a warning message.
- You cannot use the `ETCH_VS_WIDTH_AND_LENGTH` option for diffusion contacts.
- You cannot use the `ETCH_VS_WIDTH_AND_LENGTH` option together with the `ETCH_VS_CONTACT_AND_GATE_SPACINGS` option in the same `VIA` statement.
- An etch value must be less than or equal to half of the layout dimension.

Examples

In the following example, the StarRC tool sets the etch entry for length=0.045 and width=0.115 to (0.000, 0.000) because the parameter combination is invalid when the length is less than the width.

```
VIA via1{ FROM=M8 TO=M9 AREA=0.005 RPV=5
  ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY {
    LENGTHS { 0.045 0.115 0.200 }
    WIDTHS  { 0.045 0.115 }
    VALUES  { (0.015, 0.015) (0.002, 0.002) (0.003 0.001)
              (0.000, 0.000) (0.015, 0.015) (0.005 0.002)
    }
  }
}
```

See Also

- ETCH_VS_CONTACT_AND_GATE_SPACINGS
- VIA
- TC_ETCH_VS_WIDTH_AND_LENGTH

Feedback

ETCH_VS_WIDTH_AND_SPACING

Specifies conductor etch values with respect to conductor width and spacing. Valid within a CONDUCTOR block.

Syntax

```
ETCH_VS_WIDTH_AND_SPACING  {
  [RESISTIVE_ONLY | CAPACITIVE_ONLY | ETCH_FROM_TOP]
  [PARALLEL_TO_REFERENCE | PERPENDICULAR_TO_REFERENCE | PARALLEL_TO_GATE]
  [NUMBER_OF_MASKS = num_masks]
  [MASKS(a,b) [(c,d)] {
    SPACINGS { s1 s2 … sm }
    WIDTHS   { w1 w2 … wn }
    VALUES   { v(s1,w1) v(s2,w1) … v(sm,w1)
               v(s1,w2) v(s2,w2) … v(sm,w2)
               …
               v(s1,wn) v(s2,wn) … v(sm,wn)   }
   }]
  [MASKS(i,j) [(k,l)] {
    ...
   }]
}
```

Arguments

Argument	Description
RESISTIVE_ONLY	Applies etch effect to resistance only
CAPACITIVE_ONLY	Applies etch effect to capacitance only
ETCH_FROM_TOP	Specifies a trapezoidal cross section (allows the etch to affect sidewall angle)
PARALLEL_TO_REFERENCE	Applies etch effect to wires that are parallel to the reference direction
PERPENDICULAR_TO_REFERENCE	Applies etch effect to wires that are perpendicular to the reference direction
PARALLEL_TO_GATE	Applies etch on border edges that are parallel to the device gate. Use this option only for conductor layers where the LAYER_TYPE option is set to DIFFUSION.
num_masks	In a multiple mask patterning flow, the number of different mask colors
MASKS (*i*,*j*)	In a multiple mask patterning flow, the pairs of mask numbers to which the VALUES table applies

Feedback

Argument	Description
s1, *s2*, ...	Spacing values specified in ascending order; the number of spacings can be different for each set of mask pairs Units: microns
w1, *w2*, ...	Width values; the number of width values can be different for each set of mask pairs Units: microns
v(s1,w1), *v(s1,w2)*, ...	Etch values. The notation v(s1,w1) denotes the etch value corresponding to spacing s1 and width w1. Units: microns The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Description

The ETCH_VS_WIDTH_AND_SPACING command models fabricated conductor shapes. Etch and lithography effects can cause the etch to vary as a function of the width of the conductor and its spacing to neighboring features. The etch might be different on two edges of the same shape if the spacing to the nearest neighbor is different on the two sides.

To model processes with multiple etch steps, use multiple ETCH_VS_WIDTH_AND_SPACING commands. Specify them in the same order as the corresponding etch processes. Each occurrence of the ETCH_VS_WIDTH_AND_SPACING option can include more than one table if a multicolor mask flow is involved.

If you use the ETCH option in addition to one or more ETCH_VS_WIDTH_AND_SPACING tables, the ETCH_VS_WIDTH_AND_SPACING operations are applied first, followed by the ETCH operation.

For each ETCH_VS_WIDTH_AND_SPACING table, you must specify the SPACINGS and WIDTHS indexes and the VALUES matrix; these three items can appear in any order. Positive etch values cause structure widths to decrease, while negative etch values cause structure widths to increase.

The following usage notes apply:

- If the width or spacing falls between two indexes, the tool performs linear interpolation to calculate the etch value.
- If the width or spacing falls outside the table boundaries, the tool uses the nearest boundary to calculate the etch value.

Feedback

You can apply the ETCH_VS_WIDTH_AND_SPACING command to capacitance only, resistance only, or both capacitance and resistance (the default).

It is important to have correct WMIN and SMIN values for any conductor whose definition contains an ETCH_VS_WIDTH_AND_SPACING table. The WMIN and SMIN values of the conductor should be at least as large as the smallest value (the first entry) in the WIDTHS and SPACINGS tables, respectively.

Inappropriate WMIN and SMIN values might cause opens or shorts of the neighboring layers after application of the etch values in the table, as follows:

- For entries in the WIDTHS table that are equal to the WMIN value, if positive etch values are greater than or equal to half of the WMIN value, the tool issues a warning message about possible opens.
- For entries in the SPACINGS table that are equal to the SMIN value, if the absolute value of the negative etch is greater than or equal to half of the SMIN value, a potential short condition exists. However, reporting of this condition is optional because most such errors should be caught during design rule checking. To enable SMIN violation reporting, set the REPORT_SMIN_VIOLATION command to YES.

Examples

The following examples show specific applications of the ETCH_VS_WIDTH_AND_SPACING command:

- Single and Multiple Tables
- Etch From Top
- Orientation-Dependent Width
- Multiple Masks

Single and Multiple Tables

The following is an example of a single etch table:

```
CONDUCTOR metal2 {
     THICKNESS=0.65 WMIN=0.65 SMIN=0.50 RPSQ=0.62 ETCH=0.05
     ETCH_VS_WIDTH_AND_SPACING RESISTIVE_ONLY {
          SPACINGS { 0.5 0.67 0.8 }
          WIDTHS   { 0.65 0.9 }
          VALUES   { 0.1  0.05 -0.05
                     0.15 0.10 -0.10 }
     }
}
```

The following is an example of multiple etch tables:

```
CONDUCTOR M2 {
     THICKNESS=0.132 WMIN=0.050 SMIN=0.050 SIDE_TANGENT=0.06992
```

Feedback

```
        ETCH_VS_WIDTH_AND_SPACING {
        *First etch_vs_width_and_spacing table (pre ADI table)
             SPACINGS { 0.050 0.100 }
             WIDTHS   { 0.050 1.0   }
             VALUES   {-0.0027 0.0034
                        0.0003 0.0052 }
        }
        ETCH_VS_WIDTH_AND_SPACING {
        *Second etch_vs_width_and_spacing table (post ADI table)
             SPACINGS { 0.045 0.150 }
             WIDTHS   { 0.045 2.0 }
             VALUES   {-0.002 0.0014
                        0.004 -0.003 }
        }
}
```

Etch From Top

The ETCH_FROM_TOP keyword affects how the etch is applied to the structure sidewalls. Using this keyword results in a trapezoidal cross section. Figure 246 shows the effects of using negative and positive etch values along with the ETCH_FROM_TOP keyword.

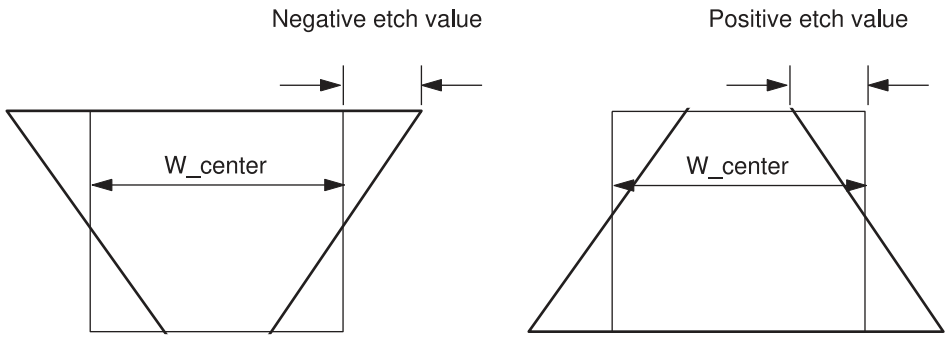
Because the amount of etch specified by the ETCH_VS_WIDTH_AND_SPACING option is based on the spacing to a neighboring structure, different sides of a conductor can be exposed to different spacings and therefore can end up with different sidewall angles after the etch. However, the center width remains constant.

The ETCH_FROM_TOP keyword takes precedence over any SIDE_TANGENT, SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING, and SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING keywords used in a conductor definition.

The resistance of a conductor is not changed by application of the ETCH_FROM_TOP keyword, because the cross sectional area of the conductor does not change. Therefore, the ETCH_FROM_TOP keyword affects only capacitance by default. You cannot use the RESISTIVE_ONLY or CAPACITIVE_ONLY keywords with the ETCH_FROM_TOP keyword.

Feedback

Figure 246 Effect of the ETCH_FROM_TOP Keyword



In the following example, a conductor of width 1 um spaced 0.5 um away from its neighbor on the left receives an etch of +0.05 um on the left edge. If that conductor is spaced 3 um away from another structure on the right side, the right edge receives an etch of -0.2 um.

```
CONDUCTOR Metal5 {
     THICKNESS=1.2 WMIN=0.3 SMIN=0.3 RPSQ = 0.62
     ETCH_VS_WIDTH_AND_SPACING ETCH_FROM_TOP {
          SPACINGS { 0.5 3 }
          WIDTHS   { 1 2 }
          VALUES   { 0.05 -0.2
                     0.1  -0.17 }
     }
}
```

Orientation-Dependent Width

Some etch processes affect lines differently depending on the orientation of the feature. To apply orientation-dependent width variation, you must specify the reference direction in either the ITF file or the StarRC command file. If the reference direction is specified in both files, the setting in the command file takes precedence.

The command formats are as follows:

- In the ITF file

  ```
  REFERENCE_DIRECTION = VERTICAL | HORIZONTAL  |  GATE
  ```

- In the StarRC command file

  ```
  REFERENCE_DIRECTION: VERTICAL | HORIZONTAL  |  NONE
  ```

One or more pairs of `ETCH_VS_WIDTH_AND_SPACING` tables can be included in each metal conductor layer. In each pair, one table must contain the `PARALLEL_TO_REFERENCE` keyword and specify the etch values for wires that are parallel to the reference direction;

Feedback

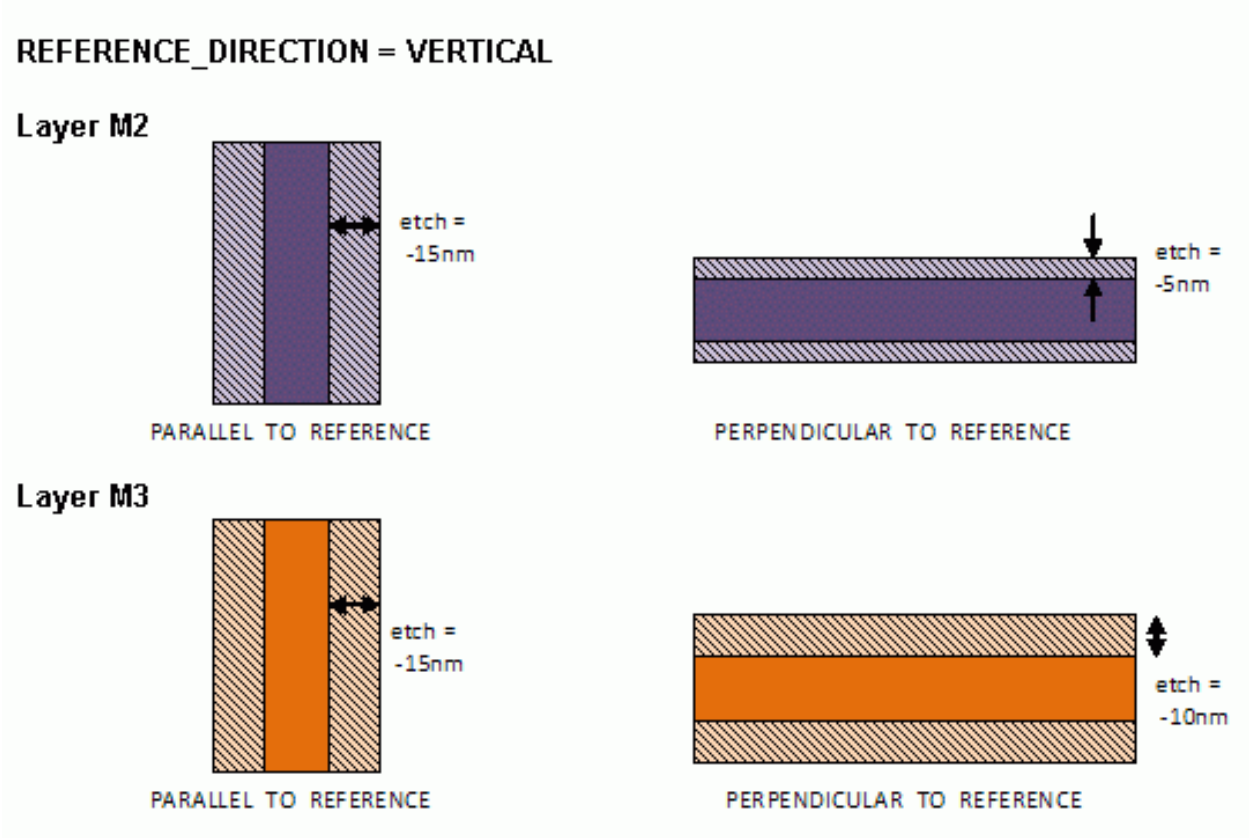
the other table must contain the PERPENDICULAR_TO_REFERENCE keyword and specify the etch values for wires that are perpendicular to the reference direction.

Note:

You cannot use the ETCH_FROM_TOP keyword with the PARALLEL_TO_REFERENCE or PERPENDICULAR_TO_REFERENCE keyword.

Figure 247 shows an example of orientation-dependent width variation in which the reference direction is vertical. The etch orientation is either parallel or perpendicular to the reference direction.

Figure 247 Example of Orientation-Dependent Width Variation



The following example shows the corresponding settings in the ITF file.

```
REFERENCE_DIRECTION = VERTICAL
…
CONDUCTOR  M2 {
  THICKNESS = …
  ETCH_VS_WIDTH_AND_SPACING
  PARALLEL_TO_REFERENCE {
    WIDTHS {0.03}    SPACINGS {0.03}    VALUES {-0.015}
```

```
  }
  ETCH_VS_WIDTH_AND_SPACING
  PERPENDICULAR_TO_REFERENCE {
    WIDTHS {0.03}    SPACINGS {0.03}    VALUES {-0.005}
  }
}
CONDUCTOR  M3 {
  THICKNESS = …
  ETCH_VS_WIDTH_AND_SPACING
  PARALLEL_TO_REFERENCE {
    WIDTHS {0.03}    SPACINGS {0.03}    VALUES {-0.015}
  }
  ETCH_VS_WIDTH_AND_SPACING
  PERPENDICULAR_TO_REFERENCE {
    WIDTHS {0.03}    SPACINGS {0.03}    VALUES {-0.010} }
}
```

Multiple Masks

A double patterning process splits features from a single layer into two separate masks. Printing the masks separately can achieve smaller features and spaces than a single mask. This practice affects extraction results if one of the masks shifts with respect to the other mask, because some features on the same layer move closer together and others move farther apart. If more than two masks are used for a single layer, the relationships are even further complicated.

You can use `ETCH_VS_WIDTH_AND_SPACING` tables to modify etch behavior based on mask shifts. The `NUMBER_OF_MASKS` keyword specifies the number of masks used to pattern the layer. The `MASKS` keyword names one or more mask pairs to which an etch table applies. A mask pair format is (self mask, neighbor mask); more than one pair can be specified for the same table. A mask ID of 0 indicates a colorless mask.

For a multiple mask process, all combinations of mask relationships must be specified. For a two-mask process, the table for mask pair (1,1) is used when adjacent features are both patterned by mask 1. Similarly, the table for mask pair (2,2) is used when adjacent features are both patterned by mask 2. The table for mask pair (1,2) is used for a feature on mask 1 when its neighbor is patterned by mask 2.

For two mask pairs with the same masks in different orders, such as mask pairs (1,2) and (2,1), the width and spacing indexes must be the same but the values can be different. If the values are also the same, you can specify one table for both pairs.

The following example uses a two-mask process. In this example, the same etch values are used for mask pair (1,2) and mask pair (2,1).

```
ETCH_VS_WIDTH_AND_SPACING
     NUMBER_OF_MASKS = 2
     MASKS (1,1) {
          SPACINGS { 0.5 0.67 0.8 }
          WIDTHS   { 0.65 0.9 }
```

```
      VALUES   { 0.1  0.05 -0.05
                 0.15 0.10 -0.10 }
      }
MASKS (1,2) (2,1) {
      SPACINGS { 0.05 0.10 }
      WIDTHS   { 0.05 0.10 }
      VALUES   { -0.002 0.0014
                  0.004 -0.003 }
      }
MASKS (2,2) {
      SPACINGS { 0.045 0.15 }
      WIDTHS   { 0.045 2.0 }
      VALUES   { -0.002 0.0014
                  0.004 -0.003}
      }
```

The following example is a table for mask ID 0 that can be optionally added to the two-mask ETCH_VS_WIDTH_AND_SPACING block to define the behavior of noncolored geometries:

```
MASKS (0,0)(0,1)(0,2)(1,0)(2,0) {
      SPACINGS { 0.05 0.10 }
      WIDTHS   { 0.005 1.0 }
      VALUES   { -0.001 0.002
                  0.004 0.003}
      }
```

Specification of tables for mask 0 is optional. If you do not define the tables, the StarRC tool derives a pessimistic evaluation from the other mask tables, as follows:

- (0,0) is set to the average of (1,2) and (2,1)
- (0,1) is set to (2,1)
- (1,0) is set to (1,2)
- (0,2) is set to (1,2)
- (2,0) is set to (2,1)
- For mask pairs between mask 0 and a mask ID bigger than 2, the tables for mask 1 and the other mask are used.

If multiple ETCH_VS_WIDTH_AND_SPACING commands are used for the same layer, the number of masks must be the same for each ETCH_VS_WIDTH_AND_SPACING command of the same type (RESISTIVE_ONLY or CAPACITIVE_ONLY). For example, you could have two CAPACITIVE_ONLY tables, each specifying two masks, and one RESISTIVE_ONLY table that does not specify any masks.

Feedback

See Also

- CONDUCTOR
- REFERENCE_DIRECTION (ITF command)
- REFERENCE_DIRECTION (StarRC command)
- SMIN
- WMIN
- REPORT_SMIN_VIOLATION

Feedback

EXTENSIONMIN

Specifies the minimum allowable extension of field poly beyond the gate polygon. Valid within a CONDUCTOR block.

Syntax

```
EXTENSIONMIN = ext_value
```

Arguments

Argument	Description
ext_value	Minimum extension value Units: microns

Description

The EXTENSIONMIN option specifies the minimum allowable extension of field poly beyond the gate polygon.

If the EXTENSIONMIN option is not specified, the WMIN value is used.

Examples

```
CONDUCTOR m1 {
     THICKNESS=1.00 WMIN=0.13 SMIN=0.15 RPSQ=0.015 EXTENSIONMIN = 0.01
}
```

See Also

- CONDUCTOR
- WMIN

Feedback

FILL_RATIO

Specifies the metal fill track utilization of emulated metal fill for use in TLUPlus files. Valid within a CONDUCTOR block.

Syntax

```
FILL_RATIO = ratio
```

Arguments

Argument	Description
ratio	Ratio of filled metal fill tracks to available metal fill tracks Maximum value: 1.0 (100 percent)

Description

The FILL_RATIO option specifies the track utilization ratio of emulated metal fill for a specified conductor layer. For emulated fill modeling, the FILL_WIDTH, FILL_SPACING, and FILL_RATIO options must all be specified.

Caution:

Fill emulation is not intended for use in a signoff flow because emulated fill does not represent the actual fill in a design.

TLUPlus models are used by the parasitic extractor in other Synopsys tools, including the IC Compiler, IC Compiler II, Fusion Compiler, and Design Compiler Topographical Mode tools.

Examples

```
CONDUCTOR m1 {
     THICKNESS=0.6 WMIN=0.3 SMIN=0.3
     FILL_RATIO = 0.4 FILL_SPACING = 1.0 FILL_WIDTH = 2.0
}
```

See Also

- FILL_SPACING
- FILL_TYPE
- FILL_WIDTH
- Emulated Metal Fill

Feedback

FILL_SPACING

Specifies the average lateral space between signal nets and emulated metal fill objects for use in TLUPlus files. Valid within a `CONDUCTOR` block.

Syntax

```
FILL_SPACING = spacing_value
```

Arguments

Argument	Description
spacing_value	Average lateral spacing between signal nets and fill objects Units: microns

Description

The `FILL_SPACING` option specifies the average lateral space between signal nets and emulated metal fill objects. If you use the `FILL_SPACING` option, you must also use the `FILL_RATIO` and `FILL_WIDTH` options. You can optionally use the `FILL_TYPE` option.

Caution:

Fill emulation is not intended for use in a signoff flow because emulated fill does not represent the actual fill in a design.

TLUPlus models are used by the parasitic extractor in other Synopsys tools, including the IC Compiler, IC Compiler II, Fusion Compiler, and Design Compiler Topographical Mode tools.

Examples

```
CONDUCTOR m1 {
     THICKNESS=0.6 WMIN=0.3 SMIN=0.3
     FILL_RATIO = 0.4 FILL_SPACING = 1.0 FILL_WIDTH = 2.0
}
```

See Also

- FILL_RATIO
- FILL_TYPE
- FILL_WIDTH
- Emulated Metal Fill

Feedback

FILL_TYPE

Specifies grounded or floating processing of emulated metal fill for use in TLUPlus files. Valid within a CONDUCTOR block.

Syntax

```
FILL_TYPE = GROUNDED | FLOATING
```

Arguments

Argument	Description
GROUNDED (default)	Models emulated metal fill as grounded fill
FLOATING	Models emulated metal fill as floating fill

Description

The FILL_TYPE option specifies whether emulated metal fill shapes are floating or grounded. This option affects only the metal fill modeled with the FILL_SPACING, FILL_WIDTH, and FILL_RATIO options.

Caution:

Fill emulation is not intended for use in a signoff flow because emulated fill does not represent the actual fill in a design.

TLUPlus models are used by the parasitic extractor in other Synopsys tools, including the IC Compiler, IC Compiler II, Fusion Compiler, and Design Compiler Topographical Mode tools.

Examples

```
CONDUCTOR m1 {
     THICKNESS=0.6 WMIN=0.3 SMIN=0.3 FILL_RATIO = 0.4
     FILL_SPACING = 1.0 FILL_WIDTH = 2.0 FILL_TYPE=FLOATING
}
```

See Also

- FILL_RATIO
- FILL_SPACING
- FILL_WIDTH
- Emulated Metal Fill

Feedback

FILL_WIDTH

Specifies the average size of emulated metal fill objects for use in TLUPlus files. Valid within a CONDUCTOR block.

Syntax

```
FILL_WIDTH = value
```

Arguments

Argument	Description
value	Average width of metal fill objects Units: microns

Description

The FILL_WIDTH option specifies the average size of emulated metal fill objects. If you use the FILL_WIDTH option, you must also use the FILL_RATIO and FILL_SPACING options. You can optionally use the FILL_TYPE option.

Caution:

Fill emulation is not intended for use in a signoff flow because emulated fill does not represent the actual fill in a design.

TLUPlus models are used by the parasitic extractor in other Synopsys tools, including the IC Compiler, IC Compiler II, Fusion Compiler, and Design Compiler Topographical Mode tools.

Examples

```
CONDUCTOR m1 {
     THICKNESS=0.6 WMIN=0.3 SMIN=0.3
     FILL_RATIO = 0.4 FILL_SPACING = 1.0 FILL_WIDTH = 2.0
}
```

See Also

- FILL_RATIO
- FILL_SPACING
- FILL_TYPE
- Emulated Metal Fill

Feedback

FROM

Specifies a conductor layer or multiple conductor layers connected by a via. Valid within a VIA block.

Syntax

```
FROM = layer | FROM {layer [layer2...]}
```

Arguments

Argument	Description
layer	Conductor layer or multiple conductor layers connected by the via

Description

The FROM option specifies the upper or lower layer (which must be a defined CONDUCTOR layer) connected by the via. The tool considers both the FROM = layer and FROM {layer...} options specified for the CONDUCTOR layer

If you specify multiple layers with the FROM option, all layers must have the same top height if they are lower layers or the same bottom height if they are upper layers. All layers must have the same LAYER_TYPE if you specify the layer type.

Examples

The following example shows how to specify a layer with the FROM = layer option:

```
VIA sub_tie {
  FROM = SUBSTRATE
  TO = M1
  AREA = 0.25
  RPV = 5
}
```

The following example shows how to specify a layer with the FROM {layer...} option:

```
VIA v2{
  FROM {fpoly1 fpoly2 gpoly1 gpoly2}
  TO = metal1
  AREA = 0.25
  RPV = 5
}
```

Feedback

See Also

- TO
- VIA

Feedback

GATE_PITCH

Specifies gate pitch values for tall contact modeling. Valid within a CONDUCTOR block.

Syntax

```
GATE_PITCH {gpmin, gp2, ..., gpmax}
```

Arguments

Argument	Description
gpmin, gp2, ..., gpmax	Typical gate pitch values, in ascending order, for table-based tall contact modeling. Must provide at least 2 values (minimum and maximum) but no more than 3. Units: microns

Description

Tall contacts that are very large or that exhibit strong coupling capacitance to other tall contacts can be modeled by using a combination of options in the CONDUCTOR and VIA definitions in the ITF file. Figure 248 illustrates the tall contact parameters used in the definitions.

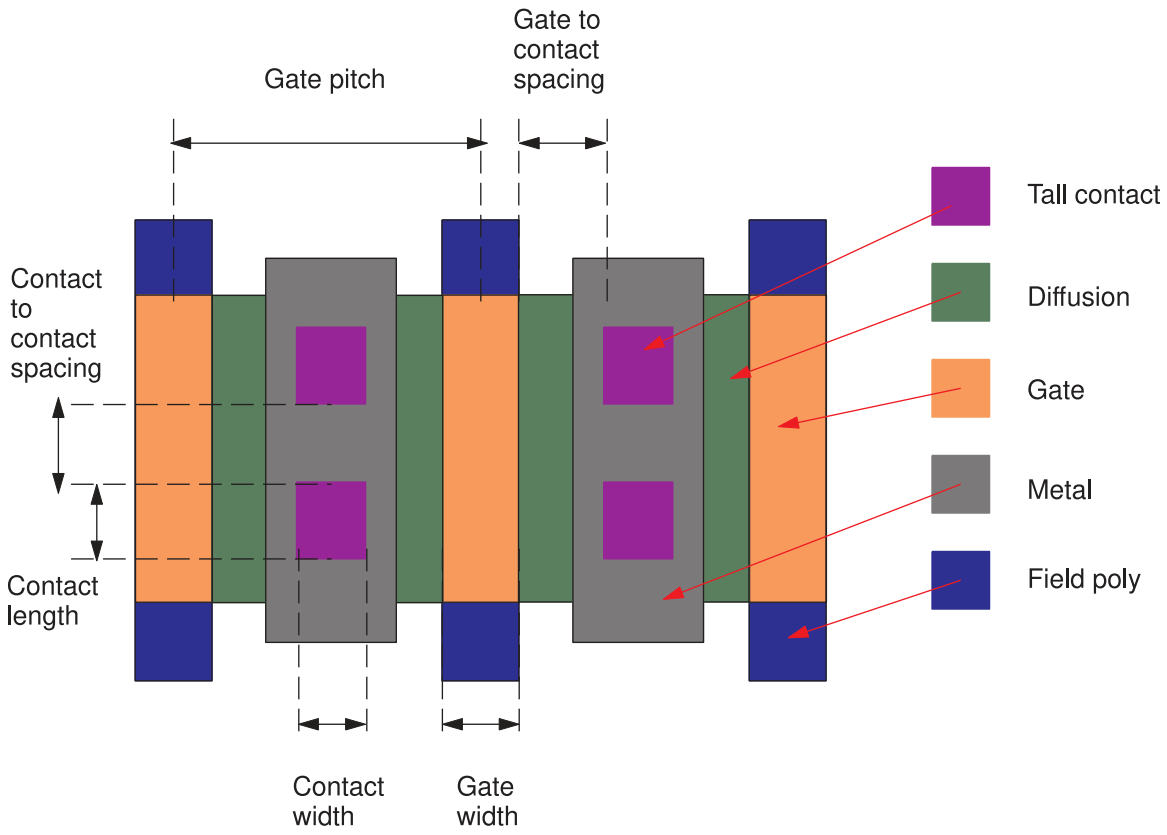
To model a tall contact with the table-based method, follow these steps:

- Include the following options in the VIA definition in the ITF file:
  - Set the LAYER_TYPE option to TALL_CONTACT.
  - (Optional) Set the DEVICE_TYPE option to a unique name. This name should be specified for one via layer and one gate conductor layer to indicate that they are used together in a tall contact structure.
  - (Optional) Include either or both of the CONTACT_WIDTH_AND_LENGTH and CONTACT_TO_CONTACT_SPACING options.
- Include the following options in the CONDUCTOR definition in the ITF file:
  - Set the LAYER_TYPE option to GATE.
  - (Optional) Set the DEVICE_TYPE option to the same name specified in the VIA definition.
  - (Optional) Include the GATE_PITCH, GATE_WIDTH, and GATE_TO_CONTACT_SPACING options.

For simultaneous multicorner (SMC) extraction, tall contact definitions must be the same for all corners.

Feedback

Figure 248 Top View of Tall Contact Layout



See Also

- CONDUCTOR
- VIA
- Table-Based Modeling of Tall Contacts

Feedback

GATE_RESISTANCE_ADJUSTMENT_FACTOR

Inserts an adjustment resistance in the gate resistance model.

Syntax

```
GATE_RESISTANCE_ADJUSTMENT_FACTOR {
   NUMBER_OF_FINS=num_of_fins {
      DISTANCE_RATIO1 {0 d2 d3... 1}
      DISTANCE_RATIO2 {0 e2 e3... 1}
      VALUES {
         f(0,0) f(0,e2) … f(0,1)
         f(d2,0) f(d2,e2) … f(d2,1)
         ...
         f(1,0) f(1,e2) … f(1,1)
      }
   }
  [NUMBER_OF_FINS=num_of_fins {
      DISTANCE_RATIO1 {0 d2 d3... 1}
      DISTANCE_RATIO2 {0 e2 e3... 1}
      VALUES {
         f(0,0) f(0,e2) … f(0,1)
         f(d2,0) f(d2,e2) … f(d2,1)
         ...
         f(1,0) f(1,e2) … f(1,1)
      }
   } ]
   GATE_LAYERS {glayer1 glayer2 ... glayerk}
}
```

Arguments

Argument	Description
num_of_fins	Number of fins
d1 d2 …e1 e2 …	Distance ratios, in ascending order, beginning with 0 and ending with 1. The two lists must be identical. Units: none
f(d,e) ...	Resistance adjustment factors, which can be positive or negative. The tables for different fin counts can have different adjustment factors. Units: none
glayer1 glayer2 ...	Names of gate layers, which must be conductor layers that contain a `LAYER_TYPE=GATE` statement.

Feedback

Description

The GATE_RESISTANCE_ADJUSTMENT_FACTOR command provides a table of adjustments to be applied to extracted gate resistance values. For this adjustment, the StarRC tool creates an additional node at the gate terminal on the gate layer and adds the adjustment resistance between the new node and the gate terminal. Resistors between other nodes and the gate terminal are moved to the new node. Capacitors remain on the gate terminal.

The adjustment resistance is calculated as follows:

1. The StarRC tool extracts Rg1, which is the resistance of the entire gate (including extensions, if any).
2. The adjustment factor from the table is multiplied by this resistance to obtain the golden gate effective resistance Rg2:

   Rg2 = Rg1 * factor
3. The adjustment resistance value is the difference between Rg2 and the resistance between the gate terminal and the via (R3):

   Rg_adj = Rg2 - R3

The adjustment is applied if all of the following conditions are true:

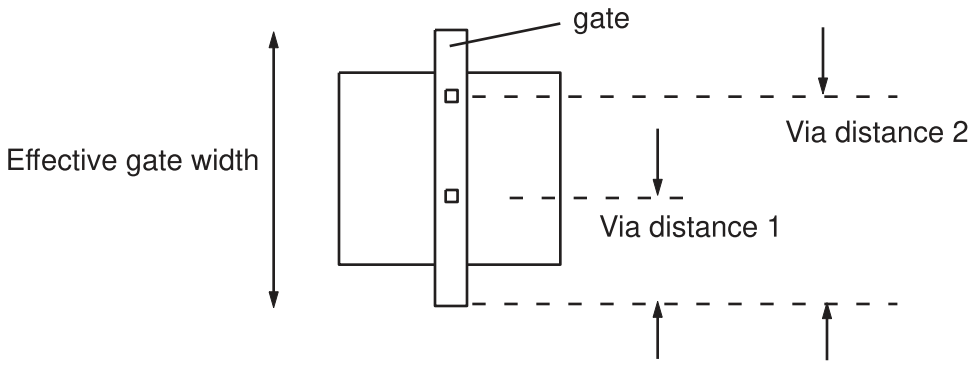
- One or more vias land on a gate or gate extension.
- The gate layer is specified in a GATE_RESISTANCE_ADJUSTMENT_FACTOR table definition.
- The ADD_GATE_ADJUSTMENT_RESISTANCE command is set to POSITIVE_ONLY (the default) or YES in the StarRC command file.

Specify one or more GATE_RESISTANCE_ADJUSTMENT_FACTOR tables at the end of the ITF file. Do not include this type of table inside a conductor definition; in this case, the grdgenxo tool issues an error message.

In Figure 249, two vias land on the gate. The distance ratios are defined as the ratio between the distance from the center of the via to the bottom of the gate and the effective gate width. The effective gate width includes gate extensions, if they exist.

Feedback

Figure 249 Dimensions for Gate Resistance Adjustment



The following usage notes apply:

- A GATE_RESISTANCE_ADJUSTMENT_FACTOR command must contain exactly one GATE_LAYER keyword, which specifies a list of one or more gate layers. The adjustment table applies to gate resistances in all of the specified gate layers. However, a single gate layer can be specified for only one GATE_RESISTANCE_ADJUSTMENT_FACTOR table.
- The GATE_LAYER keyword must appear at the end of the GATE_RESISTANCE_ADJUSTMENT_FACTOR command block.
- A GATE_RESISTANCE_ADJUSTMENT_FACTOR table cannot be specified on the same layer as a VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH table.
- For use in simultaneous multicorner extraction, the ITF files for each corner must contain GATE_RESISTANCE_ADJUSTMENT_FACTOR tables with identical indexes (the NUMBER_OF_FINS, DISTANCE_RATIO1, and DISTANCE_RATIO2 values). The adjustment values can be different for each corner.
- Requirements for the DISTANCE_RATIO1 and DISTANCE_RATIO2 lists are as follows:
  - The values in both lists must be exactly the same inside one NUMBER_OF_FINS block.
  - The values in both lists must be exactly the same inside different NUMBER_OF_FINS blocks on the same layer.
  - The values in both lists must be specified in ascending order.
  - The first value in the list must be 0 and the last value must be 1.
- The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

If the actual number of fins is between two fin counts specified in the table, the analysis applies linear interpolation. If the number of fins is smaller than the smallest specified fin count, the analysis uses the values corresponding to the smallest specified fin count. Similarly, if the number of fins is larger than the largest specified fin count, the analysis uses the values corresponding to the largest specified fin count.

You can use the `grdgenxo -res_update` command to apply a new `GATE_RESISTANCE_ADJUSTMENT_FACTOR` table to an existing nxtgrd file.

If both the `MOS_GATE_DELTA_RESISTANCE` command and the `GATE_RESISTANCE_ADJUSTMENT_FACTOR` table are specified on the same layer, the `MOS_GATE_DELTA_RESISTANCE` command is applied only if there is no via on the gate. Otherwise, the `GATE_RESISTANCE_ADJUSTMENT_FACTOR` table is applied.

In a netlist, gate adjustment resistors are reported with length 0, width 1e5, and a level equal to the gate layer ID.

Examples

```
GATE_RESISTANCE_ADJUSTMENT_FACTOR {
   NUMBER_OF_FINS = 2 {
      DISTANCE_RATIO1 {0, 0.2, 0.4, 0.6, 0.8, 1}
      DISTANCE_RATIO2 {0, 0.2, 0.4, 0.6, 0.8, 1}
      VALUES {
        1.3  2.2  3.3  4.0  4.1  3.9
        2.2  2.2  3.1  4.2  4.8  4.1
        3.3  3.1  2.3  3.4  4.2  4.0
        4.0  4.2  3.4  2.3  3.1  3.3
        4.1  4.8  4.2  3.1  2.2  2.2
        3.9  4.1  4.0  3.3  2.2  1.3
      }
   }

   NUMBER_OF_FINS = 4 {
      DISTANCE_RATIO1 {0, 0.2, 0.4, 0.6, 0.8, 1}
      DISTANCE_RATIO2 {0, 0.2, 0.4, 0.6, 0.8, 1}
      VALUES {
        1.6  3.5  6.8  8.5  9.2  6.1
        3.5  4.0  5.7  8.8  9.9  9.2
        6.8  5.7  4.2  7.7  8.8  8.5
        8.5  8.8  7.7  4.2  5.7  6.8
        9.2  9.9  8.8  5.7  4.0  3.5
        6.1  9.2  8.5  6.8  3.5  1.6
      }
   }
}
```

Feedback

See Also

- CONDUCTOR
- ADD_GATE_ADJUSTMENT_RESISTANCE

Feedback

GATE_TO_CONTACT_SMIN

Specifies the minimum spacing value between the polysilicon gate and the conductor-to-diffusion via layer. Valid within a CONDUCTOR block.

Syntax

```
GATE_TO_CONTACT_SMIN = spacing
```

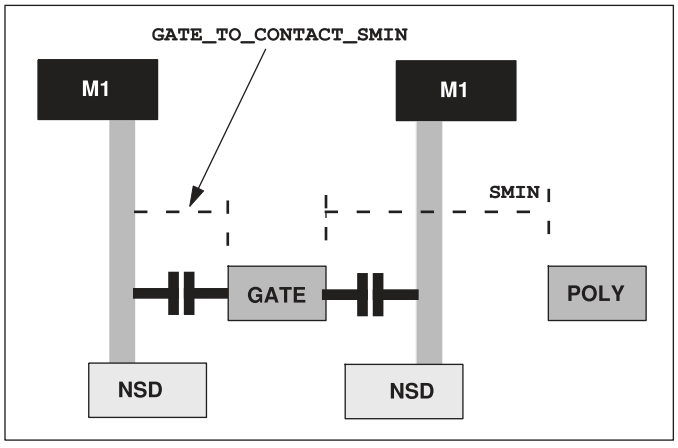
Arguments

Argument	Description
spacing	Minimum spacing between the polysilicon gate layer and the via layer between the diffusion and conductor layers Default: the SMIN value

Description

The GATE_TO_CONTACT_SMIN option specifies the minimum spacing between the polysilicon gate and the M1-to-diffusion via layer, as shown in Figure 250. Use this option with the StarRC EXTRACT_VIA_CAPS command. For the polysilicon conductors, specify both the GATE_TO_CONTACT_SMIN and SMIN values. Because gate-to-contact spacing is typically less than the SMIN value for polysilicon, specifying both values improves accuracy for EXTRACT_VIA_CAPS flows.

Figure 250 SMIN and GATE_TO_CONTACT_SMIN Definitions



Examples

```
CONDUCTOR poly {
     THICKNESS=1.00 WMIN=0.13 SMIN=0.15
```

```
        RPSQ=0.015 GATE_TO_CONTACT_SMIN=0.08
    }
```

Feedback

GATE_TO_CONTACT_SPACING

Specifies gate-to-contact spacing values for tall contact modeling. Valid within a CONDUCTOR block.

Syntax

```
GATE_PITCH {gcmin, gc2, ..., gcmax}
```

Arguments

Argument	Description
gcmin, gc2, ..., gcmax	Typical gate-to-contact spacing values, in ascending order, for table-based tall contact modeling. Must provide at least 2 values (minimum and maximum) but no more than 5. Units: microns

Description

Tall contacts that are very large or that exhibit strong coupling capacitance to other tall contacts can be modeled by using a combination of options in the CONDUCTOR and VIA definitions in the ITF file. Figure 251 illustrates the tall contact parameters used in the definitions.

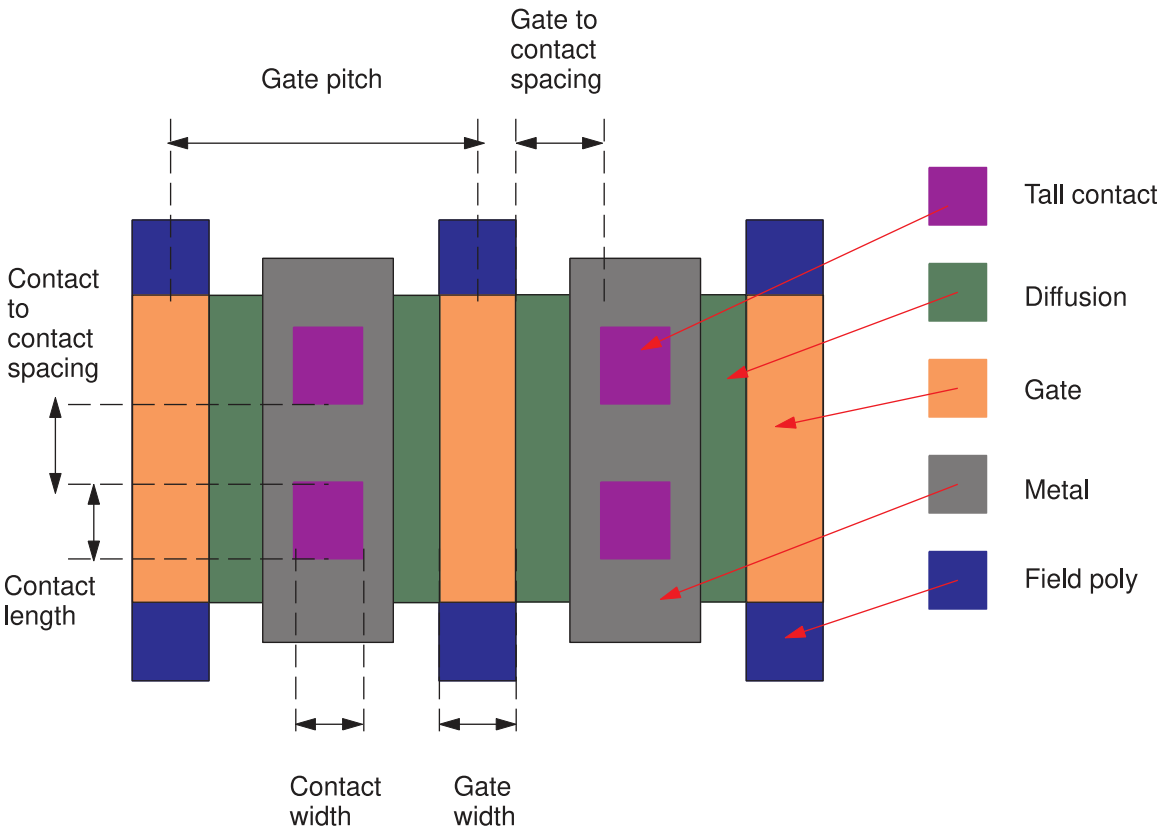
To model a tall contact with the table-based method, follow these steps:

- Include the following options in the VIA definition in the ITF file:
  - Set the LAYER_TYPE option to TALL_CONTACT.
  - (Optional) Set the DEVICE_TYPE option to a unique name. This name should be specified for one via layer and one gate conductor layer to indicate that they are used together in a tall contact structure.
  - (Optional) Include either or both of the CONTACT_WIDTH_AND_LENGTH and CONTACT_TO_CONTACT_SPACING options.
- Include the following options in the CONDUCTOR definition in the ITF file:
  - Set the LAYER_TYPE option to GATE.
  - (Optional) Set the DEVICE_TYPE option to the same name specified in the VIA definition.
  - (Optional) Include the GATE_PITCH, GATE_WIDTH, and GATE_TO_CONTACT_SPACING options.

Feedback

For simultaneous multicorner (SMC) extraction, tall contact definitions must be the same for all corners.

Figure 251 Top View of Tall Contact Layout



See Also

- CONDUCTOR
- VIA
- Table-Based Modeling of Tall Contacts

Feedback

GATE_TO_DIFFUSION_ADJUSTMENT_CAP

Models dielectric constant variations around FinFET fins. Valid in a CONDUCTOR block for gate conductor layers only.

Syntax

```
GATE_TO_DIFFUSION_ADJUSTMENT_CAP
     CHANNEL_LENGTHS {L1 L2 ... Ln}
     CHANNEL_WIDTHS  {w1 w2 w3 ... wm}
     CAPS_PER_MICRON {
                v(L1,w1) v(L2,w1) ... v(Ln,w1)
                v(L1,w2) v(L2,w2) ... v(Ln,w2)
                ...
                v(L1,wm) v(L2,wm) ... v(Ln,wm)
     }
```

Arguments

Argument	Description
L1, L2, ...	Channel lengths, in ascending order Units: microns
w1, w2, ...	Channel widths, in ascending order Units: microns
v(L1,w1)...	Capacitance adjustment for the corresponding length and width Units: fF/micron (of channel width)

Description

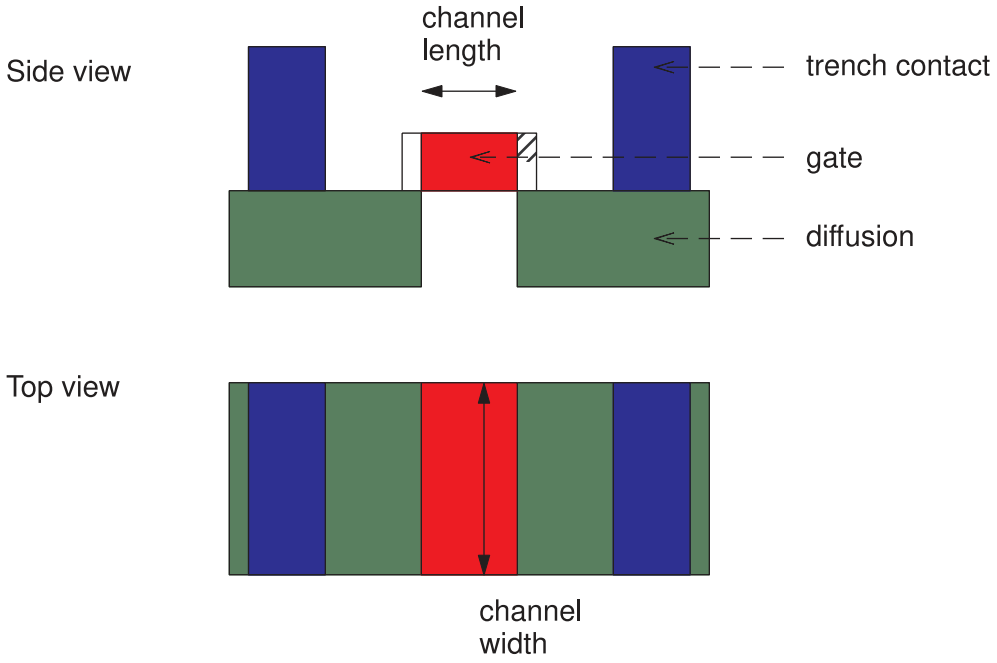
Specify this option to model the effect of variable gate-to-diffusion coupling capacitance by using an adjustment table. The dielectric properties might vary depending on FinFET channel dimensions and other process effects.

The GATE_TO_DIFFUSION_ADJUSTMENT_CAP option can only be used in conductor layers for which the LAYER_TYPE option is set to GATE.

Figure 252 shows the definitions of the channel length and width, which are determined from the layout. These values determine the capacitance adjustment, based on the GATE_TO_DIFFUSION_ADJUSTMENT_CAP table. This adjustment capacitance is placed between the device gate and diffusion and is applied in addition to other gate-to-diffusion capacitances extracted based on the MULTIGATE option.

Feedback

Figure 252 Channel Length and Channel Width Definitions



Feedback

GATE_TO_DIFFUSION_CAP

Models the gate-to-diffusion capacitance within a CONDUCTOR statement.

Syntax

```
GATE_TO_DIFFUSION_CAP {
     NUMBER_OF_TABLES = num_of_tables
     model_name1 {
          CONTACT_TO_CONTACT_SPACINGS {c1 c2 c3 … cm}
          GATE_TO_CONTACT_SPACINGS {s1 s2 s3 … sn}
          CAPS_PER_MICRON {
             v(c1,s1) v(c2,s1) … v(cm,s1)
             v(c1,s2) v(c2,s2) … v(cm,s2)
             …
             v(c1,sn) v(c2,sn) ... v(cm,sn)
          }
          GATE_WIDTHS { g1 g2 g3 ... gx }
          GATE_TO_CONTACT_SPACINGS { s1 s2 s3 ... sn }
          CAPS_PER_MICRON {
             v(g1,s1) v(g2,s1) … v(gx,s1)
             v(g1,s2) v(g2,s2) … v(gx,s2)
             …
             v(g1,sn) v(g2,sn) … v(gx,sn)
          }
     }
     ...
     model_name2 {
          …
     }
}
```

Arguments

Argument	Description
num_of_tables	Number of tables
model_name	Model name (name of the table)
c1 c2 c3 …	Nearest contact-to-contact spacing Units: microns
s1 s2 s3 …	Gate-to-contact spacings Units: microns
v(c1,s1) v(c2,s1)…	Capacitance per micron Units: femtofarads per micron
g1 g2 g3 …	Gate widths Units: microns

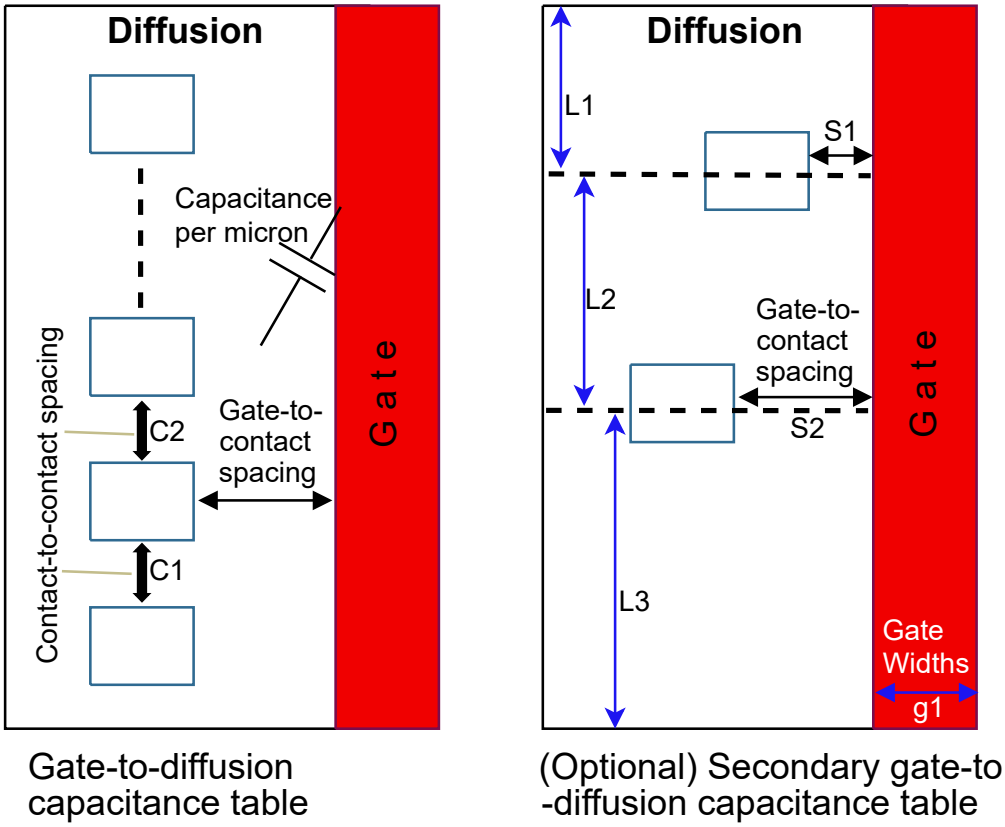
Feedback

Description

The StarRC tool can extract the gate-to-diffusion capacitance component when the `IGNORE_CAPACITANCE: ALL` setting is specified. The gate-to-diffusion intradevice capacitance is of interest because of the strong layout dependency of this capacitance component. The gate-to-contact capacitance is enabled by using the `EXTRACT_VIA_CAPS: YES` command in the command file.

To retain the gate-to-diffusion capacitance when the command file contains the `IGNORE_CAPACITANCE: ALL` command, add the `RETAIN_GATE_DIFFUSION_COUPLING` keyword to include this capacitance in the netlist. The modified command appears as `IGNORE_CAPACITANCE: ALL RETAIN_GATE_DIFFUSION_COUPLING`.

Figure 253 View of Gate-to-Diffusion Capacitance Table



Gate-to-diffusion capacitance table

(Optional) Secondary gate-to-diffusion capacitance table

The capacitance table is included as part of the gate polysilicon definition in the ITF file.

Feedback

With the gate-to-diffusion capacitance table,

- A contact etch table and gate-diffusion capacitance table cannot be individually selected. They are always selected as a set. If you need another combination of two tables for a specific type of device, define them with a new model name in the ITF file and a new database layer for the corresponding gate in the mapping file.
- The number of tables and the table names must be specified when multiple gate-to-diffusion tables are provided.
- If the GATE_TO_DIFFUSION_CAP tables are not specified in the ITF file, the tool extracts the gate-to-diffusion capacitance based on its own grdgenxo models.
- In the case where contacts do not exist, as for shared source and drain regions, the largest spacing value is taken from the specified table.
- You can add a second gate-to-diffusion capacitance table in situations such as common gates on shared diffusion (the diffusion is located in the middle of two gates that is in the multifinger device). The effect of table is inferred by gate-to-contact spacing and gate width, which is applied on top of the existing gate-to-diffusion capacitance table, on the same device model with relevant dimensions.

  In Figure 253 for the secondary gate-to-diffusion capacitance (cf), the second cf is calculated as follows:

  second cf =

$$v(g1, s1) \times L1 + \frac{v(g1, s1) + v(g1, s2)}{2} \times L2 + v(g1, s2) \times L3$$

The numbers in the CAPS_PER_MICRON field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

The generated gate-to-diffusion capacitance (cf) table is a primary table that is based on a single MOS device measurement. However, real designs mostly use MOS arrays where the gates work as a multifinger device. The voltage interaction between common gates in a multifinger device might have additional electrical charge to the shared diffusion area besides the single MOS device data provided with an ITF file. For this reason, the second gate-to-diffusion capacitance (cf) table is generated to compensate for the common gates that work on shared diffusion.

To differentiate, the existing gate-to-diffusion capacitance table is called as primary cf table and the second gate-to-diffusion capacitance table is called as the extended cf table. The effect of the second gate-to-diffusion capacitance table is decided by GATE_TO_CONTACT_SPACINGS and GATE_WIDTH, based on the existing gate-to-diffusion capacitance table of the same device with relevant dimension.

The second gate-to-diffusion capacitance (cf) table is always combined with the primary cf table, so that the second cf table is applied whenever the tool uses the primary cf table. The tool considers the following conditions to apply the second cf table:

- The availability of the second cf table from the nxtgrd file.
- The primary cf table is applied.
- The neighboring gate belongs to the same net of the current gate polygon.

Examples

Device-Dependent Gate-to-Diffusion Capacitance Table

You can specify a capacitance table based on device type. Example 41 shows the multiple gate-to-diffusion capacitance tables defined in the ITF file, one for an n-type device and another for a p-type device.

A contact etch table and a gate-diffusion capacitance table for the same type of device should have the same table name.

Example 41 Device-Dependent Gate-to-Diffusion Capacitance Table

```
CONDUCTOR gpoly {
     THICKNESS= 0.080000  WMIN= 0.040 SMIN= 0.100 RPSQ=12.000
     GATE_TO_CONTACT_SMIN=0.040 CRT1=1.924e-03 CRT2=-8.751e-07
     GATE_TO_DIFFUSION_CAP {
          NUMBER_OF_TABLES=2
          NMOS{
               CONTACT_TO_CONTACT_SPACINGS {0.08 0.12}
               GATE_TO_CONTACT_SPACINGS {0.04 0.08}
               CAPS_PER_MICRON {0.062 0.088 0.080 0.096}
          }
          PMOS {
               CONTACT_TO_CONTACT_SPACINGS {0.08 0.12}
               GATE_TO_CONTACT_SPACINGS {0.04 0.08}
               CAPS_PER_MICRON {0.088 0.1200.108 0.128}
          }
     }
}
```

Example 42 shows the first and second gate-to-diffusion capacitance tables with the gate widths for a p-type device, defined in the ITF file.

Example 42 Gate-to-Diffusion Capacitance Table With Gate Widths

```
CONDUCTOR gpoly {
     THICKNESS= 0.080000  WMIN= 0.040 SMIN= 0.100 RPSQ=12.000
     GATE_TO_CONTACT_SMIN=0.040 CRT1=1.924e-03 CRT2=-8.751e-07
     GATE_TO_DIFFUSION_CAP {
          NUMBER_OF_TABLES=1
          PMOS {
               CONTACT_TO_CONTACT_SPACINGS {0.08 0.12}
```

```
                GATE_TO_CONTACT_SPACINGS {0.04 0.08}
                CAPS_PER_MICRON {0.088 0.1200.108 0.128}
            GATE_WIDTHS {0.04 0.05 0.06}
            GATE_TO_CONTACT_SPACINGS {0.04 0.08}
            CAPS_PER_MICRON {
               0.062 0.088 0.096
               0.02 0.08 0.06
            }
         }
      }
}
```

See Also

- CAPACITIVE_ONLY_ETCH
- ETCH_VS_CONTACT_AND_GATE_SPACINGS
- IGNORE_CAPACITANCE
- GATE_TO_DIFFUSION_CHANNEL_CAP

Feedback

GATE_TO_DIFFUSION_CHANNEL_CAP

Models device-dependent gate-to-diffusion-channel capacitance for a conductor layer.

Syntax

```
CONDUCTOR gate_layer {
  LAYER_TYPE=GATE  THICKNESS= …
  GATE_TO_DIFFUSION_CHANNEL_CAP {
    NUMBER_OF_TABLES=num_of_tables
    name_of_table1 {
      CHANNEL_LENGTH  { l1 l2 l3 … }
      CHANNEL_WIDTH   { w1 w2 w3 … }
      CAPS_PER_MICRON { c(l1, w1) c(l2, w1) c(l3, w1) …
                        c(l1, w2) c(l2, w2) c(l3, w2) …
                        c(l1, w3) c(l2, w3) c(l3, w3) …
      }
    }
    name_of_table2 {
      CHANNEL_LENGTH  { l1 l2 l3 … }
      CHANNEL_WIDTH   { w1 w2 w3 … }
      CAPS_PER_MICRON { c(l1, w1) c(l2, w1) c(l3, w1) …
                        c(l1, w2) c(l2, w2) c(l3, w2) …
                        c(l1, w3) c(l2, w3) c(l3, w3) …
      }
    }
    …
  }
}
```

Arguments

Argument	Description
gate_layer	Gate layer name
num_of_tables	Number of tables
name_of_table	Name of tables
l1 l2 l3 …	Channel length Units: microns
w1 w2 w3 …	Channel width Units: microns
c(l1, w1) c(l2, w1) …	Capacitance per micron Units: femtofarads per micron

Feedback

Description

The GATE_TO_DIFFUSION_CHANNEL_CAP keyword specifies a device-dependent gate-to-diffusion-channel capacitance table. The GATE_TO_DIFFUSION_CHANNEL_CAP values depend on the CHANNEL_LENGTH and CHANNEL_WIDTH parameters. Compare this to the GATE_TO_DIFFUSION_CAP values, which depend on the GATE_TO_CONTACT_SPACINGS values.

Specify the GATE_TO_DIFFUSION_CHANNEL_CAP keyword within a CONDUCTOR statement. In the mapping file, you must provide mapping information for the different device types.

The numbers in the CAPS_PER_MICRON field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Examples

```
GATE_TO_DIFFUSION_CHANNEL_CAP {
  NUMBER_OF_TABLES = 2
  ABC_NMOS {
    CHANNEL_LENGTH {0.02 0.04 0.06}
    CHANNEL_WIDTH {0.02}
    CAPS_PER_MICRON{0.12 0.17 0.24}
  }
  XYZ_NMOS {
    CHANNEL_LENGTH {0.02 0.04 0.06}
    CHANNEL_WIDTH {0.02}
    CAPS_PER_MICRON{0.22 0.23 0.25}
  }
}
```

See Also

- CONDUCTOR
- GATE_TO_DIFFUSION_CAP
- IGNORE_GATE_CHANNEL_CAPACITANCE

Feedback

GATE_TO_LAYER_CAP

Models the gate-to-layer capacitance for a conductor layer in VFET. Valid within conductor `LAYER_TYPE = GATE`.

Syntax

```
GATE_TO_LAYER_CAP
  {
  NUMBER_OF_LAYERS = num_of_layers
  LAYER_NAME = layer_name1
    {
    GATE_LENGTHS {g1 g2 g3 … gm}
    LENGTHS {l1 l2 l3 … ln}
    WIDTHS {w1 w2 w3 … wk}
    CAPS {
          v(g1, l1, w1) v(g1, l1, w2) v(g1, l1, w3) …  v(g1, l1, wk)
          v(g1, l2, w1) v(g1, l2, w2) v(g1, l2, w3) … v(g1, l2, wk)
          …
          v(g1, ln, w1) v(g1, ln, w2) v(g1, ln, w3) … v(g1, ln, wk)
          v(g2, l1, w1) v(g2, l1, w2) v(g2, l1, w3) … v(g2, l1, wk)
          v(g2, l2, w1) v(g2, l2, w2) v(g2, l2, w3) … v(g2, l2, wk)
          …
          v(g2, ln, w1) v(g2, ln, w2) v(g2, ln, w3) … v(g2, ln, wk)
          …
          v(gm, l1, w1) v(gm, l1, w2) v(gm, l1, w3) … v(gm, l1, wk)
          …
          v(gm, ln, w1) v(gm, ln, w2) v(gm, ln, w3) … v(gm, ln, wk)
         }
    }
  LAYER_NAME = layer_name2
    {
    GATE_LENGTHS {g1 g2 g3 … gx}
    CAPS {
          v(g1) v(g2) v(g3) … v(gx)
         }
    }
  }
```

Arguments

Argument	Description
num_of_layers	Number of layers
layer_name1,*layer_name2*,...	CONDUCTOR layer name
g1 g2 g3	Gate length is the long edge of the gate Units: microns

Feedback

Argument	Description
l1 l2 l3	Length is the long edge of a layer Units: microns
w1 w2 w3	Width is the short edge of a layer Units: microns
v	Coupling capacitance of the specified pair of layers Units: femtofarads (fF)

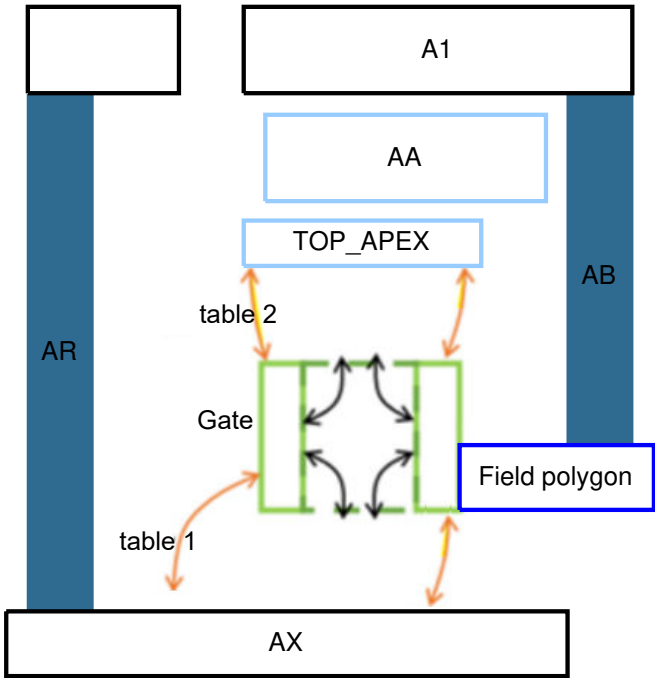
Description

A gate-to-layer capacitance table is a lookup table to calculate capacitance values between gate and its paired layers in a vertical nanosheet transistor (VFET) process. The `GATE_TO_LAYER_CAP` table is supported in StarRC and the field solver flows.

In Figure 254,

- GATE-to-TOP_APEX capacitance is looked up by the gate length and the TOP_APEX width and length
- GATE-to-AX capacitance is looked up by the gate length

Figure 254 Example of structure in VFET process



Feedback

Examples

```
CONDUCTOR GATE
  {
   THICKNESS=0.015 ...
   LAYER_TYPE=GATE ...
   GATE_TO_LAYER_CAP
    {
     NUMBER_OF_LAYERS = 2
     LAYER_NAME = TOP_APEX
     {
      GATE_LENGTHS {0.008 0.016 0.032 0.1}
      LENGTHS {0.016 0.032}
      WIDTHS {0.008 0.016}
      CAPS {
            0.1 0.12 0.14 0.16
            0.18 0.22 0.24 0.26
            0.28 0.30 0.32 0.34
            0.34 0.36 0.38 0.40
           }
     }
     LAYER_NAME = AX
     {
      GATE_LENGTHS {0.01 1.0}
      CAPS {0.015 1.5}
     }
    }
   ...
  }
```

See Also

- The Direct ITF Flow
- CONDUCTOR

Feedback

GATE_WIDTH

Specifies gate width values for tall contact modeling. Valid within a CONDUCTOR block.

Syntax

```
GATE_WIDTH {gpmin, gp2, ..., gpmax}
```

Arguments

Argument	Description
gwmin, gw2, ..., gwmax	Typical gate width values, in ascending order, for table-based tall contact modeling. Must provide at least 2 values (minimum and maximum) but no more than 5. Units: microns

Description

Tall contacts that are very large or that exhibit strong coupling capacitance to other tall contacts can be modeled by using a combination of options in the CONDUCTOR and VIA definitions in the ITF file. Figure 255 illustrates the tall contact parameters used in the definitions.

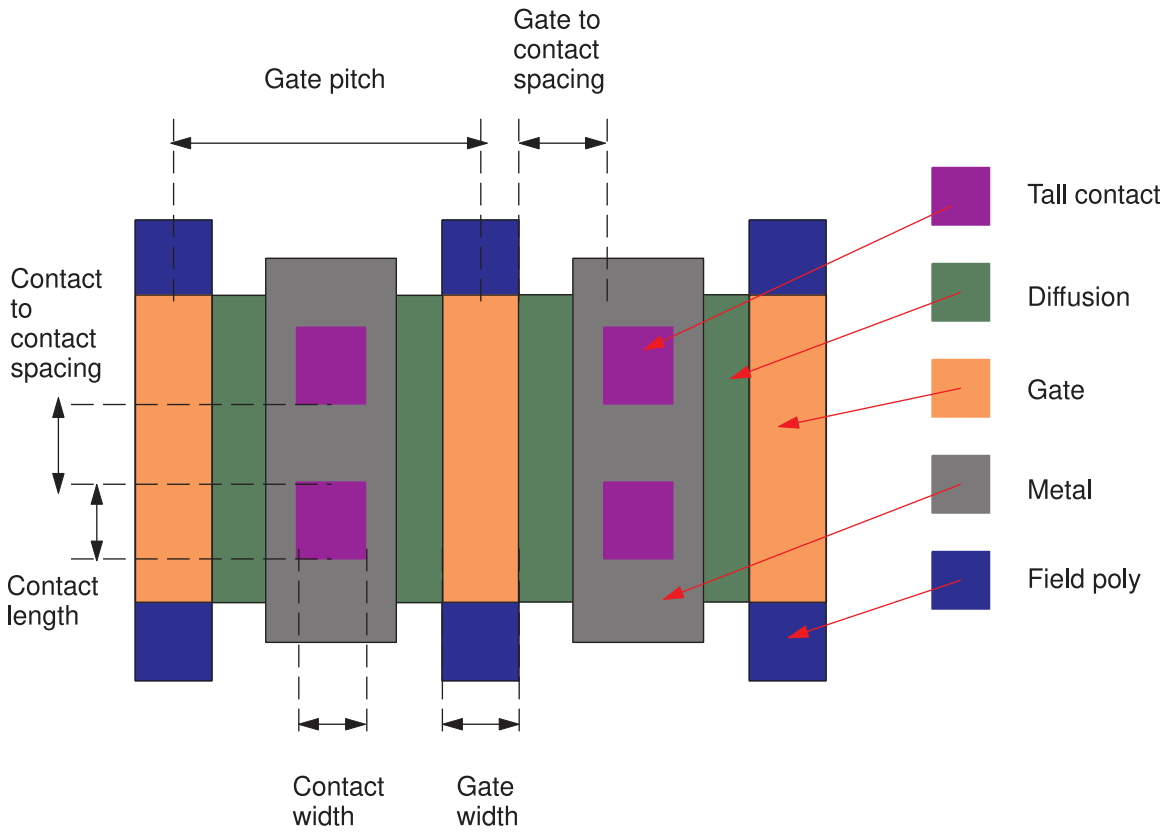
To model a tall contact with the table-based method, follow these steps:

- Include the following options in the VIA definition in the ITF file:
  - Set the LAYER_TYPE option to TALL_CONTACT.
  - (Optional) Set the DEVICE_TYPE option to a unique name. This name should be specified for one via layer and one gate conductor layer to indicate that they are used together in a tall contact structure.
  - (Optional) Include either or both of the CONTACT_WIDTH_AND_LENGTH and CONTACT_TO_CONTACT_SPACING options.
- Include the following options in the CONDUCTOR definition in the ITF file:
  - Set the LAYER_TYPE option to GATE.
  - (Optional) Set the DEVICE_TYPE option to the same name specified in the VIA definition.
  - (Optional) Include the GATE_PITCH, GATE_WIDTH, and GATE_TO_CONTACT_SPACING options.

For simultaneous multicorner (SMC) extraction, tall contact definitions must be the same for all corners.

Feedback

Figure 255 Top View of Tall Contact Layout



See Also

- CONDUCTOR
- VIA
- Table-Based Modeling of Tall Contacts

Feedback

GLOBAL_TEMPERATURE

Specifies the default nominal global temperature.

Syntax

```
GLOBAL_TEMPERATURE = temp_value
```

Arguments

Argument	Description
temp_value	Global temperature Units: degrees Celsius Default: 25

Description

The GLOBAL_TEMPERATURE statement is optional. If the GLOBAL_TEMPERATURE statement is not specified, then the nominal global temperature defaults to 25-degrees Celsius.

The nominal layer temperature overrides the global temperature when both are specified.

Examples

```
TECHNOLOGY = example_tech
GLOBAL_TEMPERATURE = 31.0
```

Feedback

HALF_NODE_SCALE_FACTOR

Shrinks the design database before extraction begins.

Syntax

```
HALF_NODE_SCALE_FACTOR = scale_factor
```

Arguments

Argument	Description
scale_factor	A positive, nonzero scale factor Default: 1.0

Description

The HALF_NODE_SCALE_FACTOR statement can optionally be included in the global parameters section of the ITF file. The HALF_NODE_SCALE_FACTOR statement directs the extraction tool to shrink the design database by the specified value before extraction begins, which is useful if you are using a half-node process technology.

When the HALF_NODE_SCALE_FACTOR value is used, the StarRC tool sets the MAGNIFY_DEVICE_PARAMS command to NO to ensure that the standard device properties ($w, $l, $area and so on) in the netlist are full-node values.

Other StarRC commands interact with the HALF_NODE_SCALE_FACTOR ITF command as shown in Table 99 and Table 100.

Table 99 Resistor Property Scaling

HALF_NODE_SCALE_FACTOR	MAGNIFICATION_FACTOR	NETLIST_UNSCALED_RES_PROP	Resistor Properties
value	value	ignored	n/a (error)
not set	value	YES	unscaled
not set	value	NO (default)	scaled
value	not set	not set (StarRC sets to YES automatically)	unscaled
value	not set	YES	unscaled
value	not set	NO	scaled

Feedback

Table 100 Coordinate Scaling in Netlist and Summary Reports

HALF_NODE_SCALE_FACTOR	MAGNIFICATION_FACTOR	NETLIST_UNSCALED_COORDINATES	Coordinates
value	value	ignored	n/a (error)
not set	value	YES	unscaled
not set	value	NO (default)	scaled
value	not set	not set (StarRC sets to YES automatically)	unscaled
value	not set	YES	unscaled
value	not set	NO	scaled

If you change the HALF_NODE_SCALE_FACTOR value, you must change the WMIN and SMIN values accordingly. The tool does not modify the WMIN or SMIN values automatically.

You can set the magnification factor in the StarRC command file after removing the value from the nxtgrd file by using the grdgenxo command, as shown in the examples. You cannot, however, delete the HALF_NODE_SCALE_FACTOR line from the nxtgrd file, because this causes the nxtgrd file to be corrupt.

Errors

If the MAGNIFICATION_FACTOR command appears in the StarRC command file and the HALF_NODE_SCALE_FACTOR command appears in the nxtgrd file, the tool issues an error message.

If the MAGNIFY_DEVICE_PARAMS:YES command is set in the StarRC command file for a run that uses an nxtgrd file containing the HALF_NODE_SCALE_FACTOR command, the tool issues a warning message stating the new value for the MAGNIFY_DEVICE_PARAMS command.

Examples

The following is an example of an ITF header using the HALF_NODE_SCALE_FACTOR option:

```
TECHNOLOGY = 65nm_example
GLOBAL_TEMPERATURE = 25
HALF_NODE_SCALE_FACTOR = 0.9

DIELECTRIC PASS2 {THICKNESS=0.800 ER=6.9}
DIELECTRIC PASS1 {THICKNESS=0.700 ER=4.0}
```

The `HALF_NODE_SCALE_FACTOR` command interacts with the `-add_sf` option of the `grdgenxo` command as follows:

- If you generated an nxtgrd file without setting the `HALF_NODE_SCALE_FACTOR` value, or you would like to change the value (for example, to 0.9), you can run the grdgenxo tool and generate an updated nxtgrd file by using the following command:

  ```
  % grdgenxo -add_sf 0.9 -i noshrink.nxtgrd -o shrink.nxtgrd
  ```

- If you generated an nxtgrd file with a `HALF_NODE_SCALE_FACTOR` value and you would like to run the StarRC tool without applying scaling, you can reset the scale factor to 1 in the nxtgrd file by using the following command:

  ```
  % grdgenxo -add_sf 1 -i shrink.nxtgrd -o noshrink.nxtgrd
  ```

See Also

- SMIN
- WMIN
- MAGNIFICATION_FACTOR
- NETLIST_UNSCALED_COORDINATES
- NETLIST_UNSCALED_RES_PROP
- Coordinate Scaling in the Netlist and Summary Reports

Feedback

ILD_VS_WIDTH_AND_SPACING

Models the microloading effect or bottom conductor thickness variation. Valid within a CONDUCTOR block.

Syntax

```
ILD_VS_WIDTH_AND_SPACING {
     DIELECTRIC_LAYER = ILD_layer_name
     SPACINGS {s1 s2 s3 ... sn}
     WIDTHS   {w1 w2 w3 ... wm}
     THICKNESS_CHANGES {
          v(s1,w1) v(s2,w1) ... v(sn,w1)
          v(s1,w2) v(s2,w2) ... v(sn,w2)
          …
          v(s1,wm) v(s2,wm) ... v(sn,wm)
     }
}
```

Arguments

Argument	Description
ILD_layer_name	The name of the dielectric layer below the conductor corresponding to the thickness variation
s1 s2 ...	Spacing in drawn dimensions
w1 w2 ...	Width in drawn dimensions
v(s1,w1) ...	Absolute thickness variation of the ILD layer. A positive value indicates an increase in thickness; a negative value indicates a decrease in thickness. Range: -0.2 to +0.2

Description

The ILD_VS_WIDTH_AND_SPACING option models the microloading effect or bottom conductor thickness variation.

Width and spacing values are drawn dimensions. The DIELECTRIC keyword specifies the dielectric layer below the conductor that exhibits the thickness variation. Each entry in the THICKNESS_CHANGES matrix specifies the thickness variation of the specified dielectric layer. A positive value indicates a thickness increase, while a negative value represents a thickness decrease.

The numbers in the THICKNESS_CHANGES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

The following requirements apply to the ILD_VS_WIDTH_AND_SPACING option:

- You can specify this option only for a dielectric layer directly below a conductor.
- You must also specify either the POLYNOMIAL_BASED_THICKNESS_VARIATION or THICKNESS_VS_DENSITY_AND_WIDTH option for the conductor layer.
- You cannot use the BOTTOM_THICKNESS_VS_SI_WIDTH option for the conductor layer.

Examples

```
ILD_VS_WIDTH_AND_SPACING {
     DIELECTRIC_LAYER = ILD3
     WIDTHS {0.1 0.2 0.3 0.4}
     SPACINGS {0.11 0.22 0.33 0.44}
     THICKNESS_CHANGES {0.130 0.134 0.138 0.140
                        0.135 0.138 0.139 0.142
                        0.138 0.139 0.139 0.143
                        0.140 0.142 0.144 0.146
     }
}
```

See Also

- BOTTOM_THICKNESS_VS_SI_WIDTH
- POLYNOMIAL_BASED_THICKNESS_VARIATION
- THICKNESS_VS_DENSITY_AND_WIDTH

Feedback

IS_CONFORMAL

Defines the material the conductor layer is deposited around and allows conformal layers to be associated with a specified conductor layer. Valid within a DIELECTRIC block.

Syntax

```
IS_CONFORMAL
```

Arguments

The IS_CONFORMAL option does not have any arguments.

Description

The IS_CONFORMAL option specifies that a dielectric is conformal. You can also use the ASSOCIATED_CONDUCTOR option to define the conductor around which the dielectric layer is deposited. If you use the IS_CONFORMAL option without the ASSOCIATED_CONDUCTOR option, the default is to measure from the top layer. If you specify a conductor, it must not be higher than the dielectric layer.

For a conformal dielectric layer, use the SW_T and TW_T statements (separately or together) to define the sidewall and topwall thicknesses around the conductor. If the TW_T or SW_T values are not specified, the defaults are 0.0. Do not use the THICKNESS option for a conformal layer.

When an ASSOCIATED_CONDUCTOR material drops with a DROP_FACTOR defined for a conductor below it, conformal dielectric layers associated with that conductor also drop.

If a conductor layer above a conformal dielectric layer overlaps with the top wall thickness of the conformal dielectric layer, the conductor cuts into the conformal dielectric layer.

Examples

```
DIELECTRIC D1 {
     IS_CONFORMAL ASSOCIATED_CONDUCTOR=met1
     SW_T=0.1 TW_T=0.1 ER=2.5
}
```

See Also

- ASSOCIATED_CONDUCTOR
- DROP_FACTOR

Feedback

IS_PLANAR

Specifies planar layers. Valid within a CONDUCTOR block.

Syntax

```
IS_PLANAR
```

Arguments

The IS_PLANAR keyword does not have arguments.

Description

Specifies that from this conductor and above, the layers do not drop because the DROP_FACTOR option is specified for the lower layers.

Examples

```
CONDUCTOR ELEC1 {
     THICKNESS = 0.010
     WMIN = 0.180
     SMIN = 0.100
     RPSQ = 0.00001
     CAPACITIVE_ONLY_ETCH = 0
     IS_PLANAR
}
```

See Also

- DROP_FACTOR

Feedback

LATERAL_CAP_SCALING_VS_SI_SPACING

Specifies a capacitance scaling factor as a function of mask level and lateral spacing. Valid within a DIELECTRIC block.

Syntax

```
LATERAL_CAP_SCALING_VS_SI_SPACING  {
   [NUMBER_OF_MASKS = num_masks]
   [MASKS(a,b) [(c,d)]] {
     (si_spacing_1, factor_1)
     (si_spacing_2, factor_2)
      ...
     (si_spacing_n, factor_n)
    }
   [MASKS(i,j) [(k,l)] {
    ...
    }]
}
```

Arguments

Argument	Description
NUMBER_OF_MASKS = *num_masks*	In a multiple mask patterning flow, the number of different mask colors.
MASKS (*a*,*b*)	In a multiple mask patterning flow, the pairs of mask numbers to which the scaling factors apply. Pair specifications (a,b) and (b,a) are equivalent. Valid entries: 0 to *num_masks*, inclusive
si_spacing_1 ... *si_spacing_n*	Spacing values specified in ascending order; the number of spacing-permittivity pairs can be different for each set of mask pairs. Units: microns Valid entries: Positive numbers greater than 0 and less than 5 times the SMIN value of the conducting layer
factor_1 ... *factor_n*	Scaling factor that corresponds to the lateral spacing value in the same pair. Units: none Valid entries: Positive numbers from 0.6 to 1.4; the last entry must be no larger than 1.0

Feedback

Description

The LATERAL_CAP_SCALING_VS_SI_SPACING statement specifies capacitance scaling factors as a function of lateral spacing for the purpose of modeling capacitance variation due to multiple patterning processes.

Use LATERAL_CAP_SCALING_VS_SI_SPACING statements in the ITF file according to the following conditions:

- No conductor mask coloring; single dielectric layer

  Use one LATERAL_CAP_SCALING_VS_SI_SPACING statement without the NUMBER_OF_MASKS and MASKS keywords.

  Alternatively, use the ER_VS_SI_SPACING statement, which does not take mask colors into account. You cannot specify both the LATERAL_CAP_SCALING_VS_SI_SPACING and ER_VS_SI_SPACING statements for the same conducting layer.

- Conductor with mask coloring and a single dielectric layer

  Use one LATERAL_CAP_SCALING_VS_SI_SPACING statement with the NUMBER_OF_MASKS and MASKS keywords.

- Conductor with mask coloring and multiple dielectric layers

  For multiple intrametal dielectric (IMD) layers that share the same conducting layer, a LATERAL_CAP_SCALING_VS_SI_SPACING table can be specified on some or all of the IMD layers.

  All LATERAL_CAP_SCALING_VS_SI_SPACING tables for dielectrics associated with the same conductor must have the same spacing values for a specific mask pair.

  All of the tables associated with a single conductor must use the same NUMBER_OF_MASKS value. In addition, if a mask-based ETCH_VS_WIDTH_AND_SPACING statement is used for the same conductor layer, its NUMBER_OF_MASKS value must be the same.

Spacing values are silicon (post-etch) spacing values. If the actual spacing falls within two values in the table, linear interpolation is used to calculate the scaling factor.

The NUMBER_OF_MASKS keyword specifies the number of masks used to pattern the layer. The MASKS keyword names one or more mask pairs to which an etch table applies. A mask pair format is (self mask, neighbor mask); more than one pair can be specified for the same table. A mask ID of 0 indicates a colorless mask.

When the NUMBER_OF_MASKS keyword is used, at least one colored mask combination must be specified with the MASKS keyword. For mask pairs that require no scaling, do not specify them; in this case, the StarRC tool sets the factor to 1.

Feedback

Specifying mask pairs that include mask 0 is optional. If they are missing, the tool maps mask pairs (0,1), (1,0), (0,2), and (2,0) to mask pair (1,2). For larger mask numbers, the tool maps mask pairs (0,n) and (n,0) map to mask pair (1,n). If mask pair (1,n) is not specified, the tool uses a scaling factor of 1. You cannot specify the same mask pair more than one time.

The `DAMAGE_THICKNESS` option, if present, has higher priority.

Errors

The `NUMBER_OF_MASKS` and `MASKS` keywords must be used if the design database contains colored mask information; the keywords cannot be used if the masks are uncolored. If a mismatch occurs, the StarRC tool issues an error message and exits the run.

The `LATERAL_CAP_SCALING_VS_SI_SPACING` statement is not valid for the following layers:

- Conformal dielectric layers
- Dielectric layers whose associated conductor layer contains a `DROP_FACTOR` specification

Examples

The following example provides a single table for uncolored masks:

```
LATERAL_CAP_SCALING_VS_SI_SPACING {
          (0.1,0.7) (0.12,0.8) (0.15,0.95) (0.2,1.0) }
```

The following example uses different values for different mask pairs. .

```
LATERAL_CAP_SCALING_VS_SI_SPACING {
    NUMBER_OF_MASKS = 2
    MASKS (1,2) {
      (0.05 0.7) (0.08 0.8) (0.1 1.0)
    }
    MASKS (1,1) (2,2) {
      (0.05 1.25)
      (0.08 1.17)
      (0..1 1.0)
    }
}
```

See Also

- ER_VS_SI_SPACING
- ETCH_VS_WIDTH_AND_SPACING

Feedback

LATERAL_CAP_SCALING_VS_SPACING

Specifies a capacitance scaling factor as a function of mask level and lateral spacing. Valid within a DIELECTRIC block.

Syntax

```
LATERAL_CAP_SCALING_VS_SPACING  {
   [NUMBER_OF_MASKS = num_masks]
   [MASKS(a,b) [(c,d)]] {
     (spacing_1, factor_1)
     (spacing_2, factor_2)
      ...
     (spacing_n, factor_n)
    }
   [MASKS(i,j) [(k,l)] {
    ...
    }]
}
```

Arguments

Argument	Description
NUMBER_OF_MASKS = *num_masks*	In a multiple mask patterning flow, the number of different mask colors.
MASKS (*a*,*b*)	In a multiple mask patterning flow, the pairs of mask numbers to which the scaling factors apply. Pair specifications (a,b) and (b,a) are equivalent. Valid entries: 0 to *num_masks*, inclusive
spacing_1 ... *spacing_n*	Spacing values specified in ascending order; the number of spacing-permittivity pairs can be different for each set of mask pairs. Units: microns Valid entries: Positive numbers greater than 0 and less than 5 times the SMIN value of the conducting layer
factor_1 ... *factor_n*	Scaling factor that corresponds to the lateral spacing value in the same pair. Units: none Valid entries: Positive numbers from 0.6 to 1.4; the last entry must be no larger than 1.0

Feedback

Description

The LATERAL_CAP_SCALING_VS_SPACING statement specifies capacitance scaling factors as a function of lateral spacing for the purpose of modeling capacitance variation due to multiple patterning processes.

Use LATERAL_CAP_SCALING_VS_SPACING statements in the ITF file according to the following conditions:

- No conductor mask coloring; single dielectric layer

  Use one LATERAL_CAP_SCALING_VS_SPACING statement without the NUMBER_OF_MASKS and MASKS keywords.

  Alternatively, use the ER_VS_SI_SPACING statement, which does not take mask colors into account. You cannot specify both the LATERAL_CAP_SCALING_VS_SPACING and ER_VS_SI_SPACING statements for the same conducting layer.

- Conductor with mask coloring and a single dielectric layer

  Use one LATERAL_CAP_SCALING_VS_SPACING statement with the NUMBER_OF_MASKS and MASKS keywords.

- Conductor with mask coloring and multiple dielectric layers

  For multiple intrametal dielectric (IMD) layers that share the same conducting layer, a LATERAL_CAP_SCALING_VS_SPACING table can be specified on some or all of the IMD layers.

  All LATERAL_CAP_SCALING_VS_SPACING tables for dielectrics associated with the same conductor must have the same spacing values for a specific mask pair.

  All of the tables associated with a single conductor must use the same NUMBER_OF_MASKS value. In addition, if a mask-based ETCH_VS_WIDTH_AND_SPACING statement is used for the same conductor layer, its NUMBER_OF_MASKS value must be the same.

Spacing values are drawn spacing values. If the actual spacing falls within two values in the table, linear interpolation is used to calculate the scaling factor.

The NUMBER_OF_MASKS keyword specifies the number of masks used to pattern the layer. The MASKS keyword names one or more mask pairs to which an etch table applies. A mask pair format is (self mask, neighbor mask); more than one pair can be specified for the same table. A mask ID of 0 indicates a colorless mask.

When the NUMBER_OF_MASKS keyword is used, at least one colored mask combination must be specified with the MASKS keyword. For mask pairs that require no scaling, do not specify them; in this case, the StarRC tool sets the factor to 1.

Feedback

Specifying mask pairs that include mask 0 is optional. If they are missing, the tool maps mask pairs (0,1), (1,0), (0,2), and (2,0) to mask pair (1,2). For larger mask numbers, the tool maps mask pairs (0,n) and (n,0) map to mask pair (1,n). If mask pair (1,n) is not specified, the tool uses a scaling factor of 1. You cannot specify the same mask pair more than one time.

The DAMAGE_THICKNESS option, if present, has higher priority.

Errors

The NUMBER_OF_MASKS and MASKS keywords must be used if the design database contains colored mask information; the keywords cannot be used if the masks are uncolored. If a mismatch occurs, the StarRC tool issues an error message and exits the run.

The LATERAL_CAP_SCALING_VS_SPACING statement is not valid for the following layers:

- Conformal dielectric layers
- Dielectric layers whose associated conductor layer contains a DROP_FACTOR specification

Examples

The following example provides a single table for uncolored masks:

```
LATERAL_CAP_SCALING_VS_SPACING {
          (0.1,0.7) (0.12,0.8) (0.15,0.95) (0.2,1.0) }
```

The following example uses different values for different mask pairs. .

```
LATERAL_CAP_SCALING_VS_SPACING {
    NUMBER_OF_MASKS = 2
    MASKS (1,2) {
      (0.05 0.7) (0.08 0.8) (0.1 1.0)
    }
    MASKS (1,1) (2,2) {
      (0.05 1.25)
      (0.08 1.17)
      (0..1 1.0)
    }
}
```

See Also

- ER_VS_SI_SPACING
- ETCH_VS_WIDTH_AND_SPACING

Feedback

LAYER_TYPE

Specifies the layer type within a CONDUCTOR or VIA block.

Syntax

```
LAYER_TYPE = GATE | FIELD_POLY | DIFFUSION | BUMP | TRENCH_CONTACT
             | TALL_CONTACT | CAPACITOR | RESISTOR | ROUTING_VIA
```

Arguments

Argument	Description
GATE	A conducting layer that forms the gate of a device. If separate ITF conducting layers are not specified for gate and field polysilicon, specify the combined gate-field polysilicon layer as LAYER_TYPE = GATE. Valid only for CONDUCTOR layers.
FIELD_POLY	A field polysilicon layer outside of the device region. Valid only for CONDUCTOR layers.
DIFFUSION	A layer used for source or drain diffusion regions. Valid only for CONDUCTOR layers.
BUMP	A pseudo-metal via layer, used to connect microbumps to the top metal layer. Valid only for CONDUCTOR layers.
TRENCH_CONTACT	A conducting layer used for trench contacts. This includes both M1-to-diffusion trench contacts and M1-to-polysilicon trench contacts that can be used both inside and outside of the device region. Valid only for CONDUCTOR layers.
TALL_CONTACT	A layer for vias through thick layers. Valid only for VIA layers.
CAPACITOR	A special-purpose layer for interleaved metal-finger capacitor structures. Valid only for CONDUCTOR layers.
RESISTOR	A special-purpose layer for resistor devices, usually having differentiating physical parameters such as a thin thickness. Valid only for CONDUCTOR layers.
ROUTING_VIA	A layer that connects a routing conductor layer and a device conductor layer (such as a field poly or trench contact layer). Extraction conditions are determined by the layers being connected. Valid only for CONDUCTOR layers.

Description

For advanced process technologies, information in the ITF file about the function of the conducting or via layers improves capacitance extraction accuracy. You can optionally include a LAYER_TYPE keyword to guide the analysis.

Feedback

Note the following constraints on the relative vertical position of conductors in the interconnect stack:

- Conductors with `LAYER_TYPE = GATE` and `LAYER_TYPE = FIELD_POLY` must be covertical.
- Conductors with `LAYER_TYPE = TRENCH_CONTACT` must be covertical with or above conductors with `LAYER_TYPE = GATE` or `LAYER_TYPE = FIELD_POLY`.
- Conductors with `LAYER_TYPE = DIFFUSION` must be below the conductors with `LAYER_TYPE = GATE`.

Errors

If the constraints on the layer type are not satisfied, the grdgenxo tool issues an error message.

Examples

The following example uses the `LAYER_TYPE` option to identify gate and diffusion layers:

```
CONDUCTOR PS {
  THICKNESS = 0.04 WMIN = 0.04 SMIN = 0.04
  GATE_TO_CONTACT_SMIN = 0.02
  LAYER_TYPE = GATE
  …
}
DIELECTRIC DP1 {
  THICKNESS = 0.001
  …
}
DIELECTRIC D_DIFF {
  THICKNESS = 0.04
  …
}
CONDUCTOR DIFF {
  THICKNESS = 0.04 WMIN = 0.04 SMIN = 0.04
  LAYER_TYPE = DIFFUSION
  …
}
```

See Also

- CONDUCTOR
- VIA

Feedback

LINE_END_EXTENSION_TABLE

Specifies conductor line-end extensions. Valid within a CONDUCTOR block.

Syntax

```
LINE_END_EXTENSION_TABLE
  PARALLEL_TO_REFERENCE | PERPENDICULAR_TO_REFERENCE | ART = value
  [VIA_ENCLOSURE_DISTANCE = distance]
  [NUMBER_OF_MASKS = num_masks]
  [MASKS(a,b) [(c,d)] {
    NUMBER_OF_WIDTHS = num_widths
    WIDTH1 = widthA {
      SPACINGS { s1 s2 … sm }
      WIDTH2S  { w1 w2 … wn }
      VALUES   { v(s1,w1) v(s2,w1) … v(sm,w1)
                 v(s1,w2) v(s2,w2) … v(sm,w2)
                 …
                 v(s1,wn) v(s2,wn) … v(sm,wn) }  }
    WIDTH1 = widthB { ... }
  }
  [MASKS(i,j) [(k,l)] { ... }]
}
```

Arguments

Argument	Description
PARALLEL_TO_REFERENCE	Applies to wires that are parallel to the reference direction
PERPENDICULAR_TO_REFERENCE	Applies to wires that are perpendicular to the reference direction
ART	Specifies if the edge is applied with LINE_END_EXTENSION_TABLE option or ETCH_VS_WIDTH_AND_SPACING option
VIA_ENCLOSURE_DISTANCE	Specify a value for the enclosure distance from a metal edge
num_masks	In a multiple mask patterning flow, the number of different mask colors Range: 1 to 20
MASKS(*a*,*b*), (*c*,*d*), ...	In a multiple mask patterning flow, the pairs of mask numbers to which the extension table applies
num_widths	The number of conductor widths for which a table is provided
widthA, widthB, ...	The conductor width to which the subsequent table applies

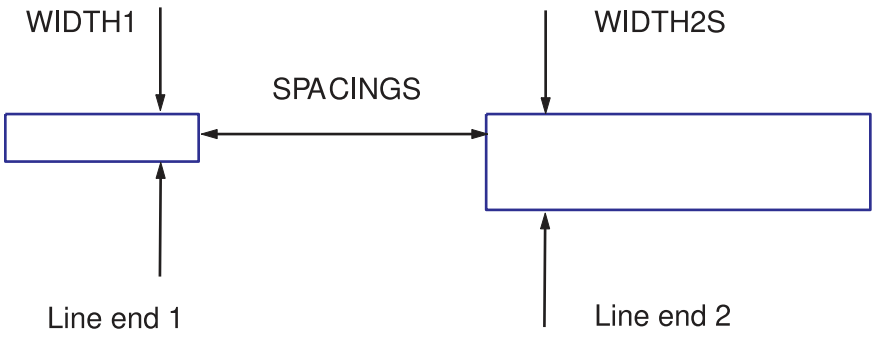
Feedback

Argument	Description
s1, *s2*, ...	Spacing values specified in ascending order; the values must be the same for all tables Units: microns
w1, *w2*, ...	Extension width values specified in ascending order; the values must be the same for all tables Units: microns
v(s1,w1), *v(s1,w2)*, ...	Extension values. The notation v(s1,w1) denotes the value corresponding to spacing s1 and width w1. Units: microns

Description

A line-end extension table is a set of lookup tables that modifies drawn conductor dimensions before extraction. Figure 256 shows two line ends and the geometric parameters represented by the WIDTH1, WIDTH2S, and SPACINGS keywords. For each specified width of conductor line end 1, the table provides effective line extension values as a function of the distance to line end 2 and the width of line end 2.

Figure 256 Line End Extension Parameters

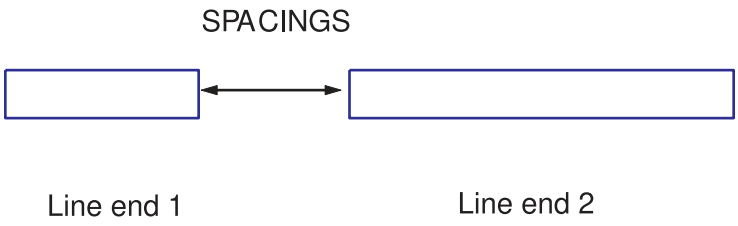


You can optionally use the CUT_END_EXTENSION_TABLE statement to specify a one-dimensional table that provides line extension values as a function of the spacing between two lines, as illustrated in Figure 257. If present, the CUT_END_EXTENSION_TABLE option is applied before the LINE_END_EXTENSION_TABLE option.

The maximum spacing in the CUT_END_EXTENSION_TABLE option is usually smaller than the spacings in the LINE_END_EXTENSION_TABLE option.

Feedback

Figure 257 Cut End Extension Parameters



The following usage notes apply:

- A single conductor block can contain at most one `LINE_END_EXTENSION_TABLE` definition.
- The ITF file must contain a `REFERENCE_DIRECTION` statement.
- The `LINE_END_EXTENSION_TABLE` definition must include either the `PARALLEL_TO_REFERENCE` or `PERPENDICULAR_TO_REFERENCE` keyword.
- If a conductor block contains both a `LINE_END_EXTENSION_TABLE` option and an `ETCH_VS_WIDTH_AND_SPACING` option, one of the options must include the `PARALLEL_TO_REFERENCE` keyword and the other must include the `PERPENDICULAR_TO_REFERENCE` keyword.
- For simultaneous multicorner runs, all corners must contain the same `LINE_END_EXTENSION_TABLE` definition.
- Line end extension tables do not apply to non-Manhattan polygons.
- The syntax items must be specified exactly in the order shown.

For multiple mask patterning flows, you can specify different extension tables for different mask pairs. The following usage notes apply:

- Every mask pair must have a table definition. If you do not want to apply an extension table for a specific mask pair, you must specify the extension table syntax for that mask pair and set all of the values to 0. Tables that contain all zeros must use the same spacing and width indexes as the other tables.
- Noncolor polygons in a colored design are assigned to `MASK=0`. If you do not explicitly define mask pairs that include a mask index of 0 in the `LINE_END_EXTENSION_TABLE` definition, the StarRC tool does not apply an extension table to noncolor polygons or to polygons that interact with noncolor polygons.
- If there is only one mask, all mask pairs must be defined, including pairs with mask 0.

Feedback

Applying Line-End Extension Blockage Based on Enclosure Distance of Via

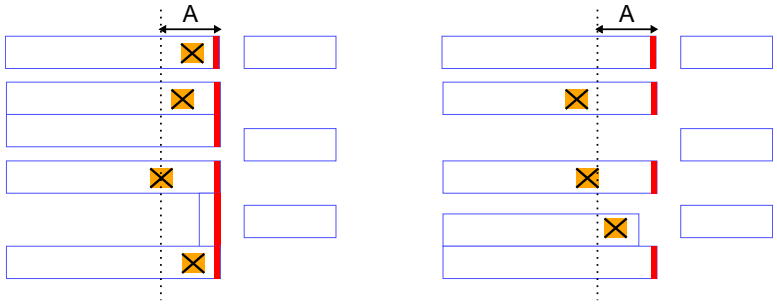
To apply the conductor line-end extensions based on LINE_END_EXTENSION_TABLE, specify the VIA_ENCLOSURE_DISTANCE keyword with the LINE_END_EXTENSION_TABLE option. The tool determines to apply the conductor line-end extensions based on the enclosure distance of a via from the metal edge, as follows:

- If a via is within a specified enclosure distance from the metal edge, the line-end extension is applied
- If a via is not within a specified enclosure distance from the metal edge, the line-end extension is not applied

The following usage notes apply:

- If the VIA_ENCLOSURE_DISTANCE keyword is not used with the LINE_END_EXTENSION_TABLE option, the tool applies line-end extension without checking whether the via exists.
- If the VIA_ENCLOSURE_DISTANCE keyword is used, the tool checks vias on both the upper and lower via layers of the metal edge.
- If any via overlaps (abut polygons not included) with the area of VIA_ENCLOSURE_DISTANCE (A in Figure 258) and the line-end extension edge (red line in Figure 258), the line-end extension is applied based on LINE_END_EXTENSION_TABLE. Otherwise, the line-end extension is not applied.

Figure 258 Via Overlapping With the Area of VIA_ENCLOSURE_DISTANCE and the Line-End Extension Edge



Examples

The following example applies both cut end extensions and line end extensions for a process without multiple masks.

```
CONDUCTOR MetalX {
...
CUT_END_EXTENSION_TABLE PERPENDICULAR_TO_REFERENCE {
   SPACINGS {0.03}
```

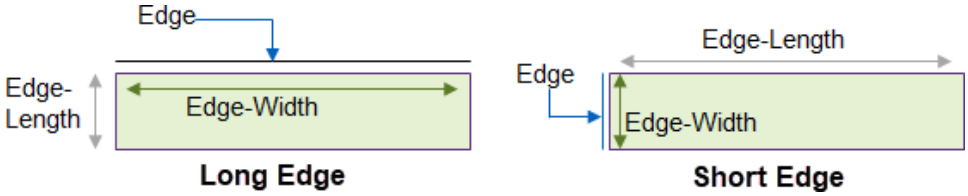
```
   VALUES {0.002}   }
LINE_END_EXTENSION_TABLE PERPENDICULAR_TO_REFERENCE {
   NUMBER_OF_WIDTHS = 2
   WIDTH1 = 0.02 {
      SPACINGS {0.08 0.100 0.200}
      WIDTH2S {0.05 0.08 0.200}
      VALUES {0.00 0.00 0.00 0.00 0.00 0.001 0.002 0.003 0.004}
   }
   WIDTH1 = 0.04 {
      SPACINGS {0.08 0.100 0.200}
      WIDTH2S {0.05 0.08 0.200}
      VALUES {0.00 0.001 0.002 0.004 0.005 0.008 0.010 0.012 0.015}
   }
}
```

Applying Line-End Extension With Aspect Ratio Threshold

To apply the aspect ratio threshold, specify the `ART` keyword with the `LINE_END_EXTENSION_TABLE` option. The tool determines whether an edge should be applied with LEE or the etching (`ETCH_VS_WIDTH_AND_SPACING`) options. The ART option replaces the `PARALLEL_TO_REFERENCE` and `PERPENDICULAR_TO_REFERENCE` options which enables the orientation.

The aspect ratio is calculated as follows, where Edge-Length is the length of line-end polygon and Edge-Width is the width of line-end edge.

```
Aspect Ratio (AR) = Edge-Length / Edge-Width
```



When the Edge AR is greater than or equal to the ART value, the Edge is applied with `LINE_END_EXTENSION_TABLE` option, otherwise it is applied with `ETCH_VS_WIDTH_AND_SPACING` option.

Examples

The following example specifies the ART value.

```
Conductor metal1 {
   LINE_END_EXTENSION_TABLE ART=art {
      NUMBER_OF_WIDTHS = nw
      WIDTH1=w1 {
      SPACINGS { s1 s2 … sm }
      LENGTH1S { L1 L2 … Ln }
      VALUES   { v(w1,s1,L1) v(w1,s2,L1) … v(w1,sm,L1)
                 v(w1,s1,L2) v(w1,s2,L2) … v(w1,sm,L2)
                 …
```

```
                    v(w1,s1,Ln) v(w1,s2,Ln) … v(w1,sm,Ln)
       }
       }
    …
    WIDTH1 = wk {
       SPACINGS { s1 s2 … sm }
       LENGTH1S   { L1 L2 … Ln}
       VALUES {
           v(wk,s1,L1) v(wk,s2,L1) … v(wk,sm,L1)
           v(wk,s1,L2) v(wk,s2,L2) … v(wk,sm,L2)
           …
           v(wk,s1,Ln) v(wk,s2,Ln) … v(wk,sm,Ln)
       }
       }
    }
 }
```

Specifying the Length of the Line-end Extension Tables

A line-end extension table shows two line ends and the geometric parameters represented by the `LENGTH1S` keyword. The `CUT_END_EXTENSION_TABLE` is assumed to specify a one-dimensional table that provides line extension values as a function of the spacing between two lines.

The tool determines to apply the `LENGTH1S` on the polygon based on the length selection, as follows:

- Select the shortest distance when multiple polygon lengths are available.
- Select the shortest length when polygon connects `skip_cell` or `skip_pcell` polygons
- Select the combined length If top polygon contacts the skip cell polygon and the shapes are aligned.

Syntax

```
LINE_END_EXTENSION_TABLE PARALLEL_TO_REFERENCE {
      NUMBER_OF_WIDTH1S=nw
      WIDTH1=w1 {
      SPACINGS { s1 s2 … sm }
      LENGTH1S { L1 L2 … Ln }
      VALUES   { v(w1,s1,L1) v(w1,s2,L1) … v(w1,sm,L1)
                 v(w1,s1,L2) v(w1,s2,L2) … v(w1,sm,L2)
                 …
                 v(w1,s1,Ln) v(w1,s2,Ln) … v(w1,sm,Ln)
      }
      }
    …
    WIDTH1 = wk {
       SPACINGS { s1 s2 … sm }
       LENGTH1S   { L1 L2 … Ln}
       VALUES {
```

```
            v(wk,s1,L1) v(wk,s2,L1) … v(wk,sm,L1)
            v(wk,s1,L2) v(wk,s2,L2) … v(wk,sm,L2)
            …
            v(wk,s1,Ln) v(wk,s2,Ln) … v(wk,sm,Ln)
        }
        }
    }
 }
CUT_END_EXTENSION_TABLE  PERPENDICULAR_TO_REFERENCE {
  SPACINGS { s1 … sm }
  VALUES {
     v(s1) v(s2) … v(sm)
  }
}
```

The StarRC tool issues an error message if `LENGTH1S` and `WIDTH2S` are both defined in line-end extension table.

See Also

- CUT_END_EXTENSION_TABLE
- conducting_layers

Feedback

LINKED_TO

Specifies a conductor layer that is a candidate to be considered capacitively identical. Valid within a `CONDUCTOR` block.

Syntax

```
LINKED_TO = layer_name
```

Arguments

Argument	Description
layer_name	Conductor layer name

Description

The `LINKED_TO` option specifies the name of another conductor layer that is a candidate to be considered capacitively identical to this layer.

To save runtime, when the `grdgenxo -link_device_models` command is used, the grdgenxo tool attempts to identify conductors that have identical capacitive properties to reuse results previously generated with matching layers.

Use the `LINKED_TO` option option as follows:

- If the nxtgrd file is generated using only ITF commands, the `LINKED_TO` command is not necessary.
- If the nxtgrd file is generated with both ITF and QTF commands, you can use your knowledge of the process to refine the search for a match. If any `LINKED_TO` statements appear in a device-related conductor definition, automatic linking in device models is disabled and only the explicitly linked conductor is a candidate to be considered identical.

  Linking occurs only if the layers are capacitively equivalent. If you use the `LINKED_TO` statement to specify a layer that is not capacitively equivalent to the primary conductor layer, the StarRC tool issues a warning message and does not link the layers.

Examples

```
CONDUCTOR gate_A {
     LINKED_TO = gate_B  }
```

Feedback

See Also

- CONDUCTOR
- The grdgenxo Command
- The QTF Flow

Feedback

MEASURED_FROM

Modifies the reference location where thickness is measured. Valid within a CONDUCTOR or DIELECTRIC block.

Syntax

```
MEASURED_FROM = dielectric_layer | TOP_OF_CHIP
```

Arguments

Argument	Description
dielectric_layer	The name of the dielectric layer from which the measurement is made. This layer name must be defined in the ITF file. The default is the dielectric layer immediately below.
TOP_OF_CHIP	Valid only within a DIELECTRIC block to facilitate the creation of conformal dielectrics. Creates the bottom plane from the layers already present below the new layer and mimics the topology of the existing base.

Description

The MEASURED_FROM option modifies the reference location where thickness is measured to account for process characteristics such as conformal dielectrics, mixed conformal and planar dielectrics, and covertical conductors.

When used with a CONDUCTOR layer definition, the MEASURED_FROM keyword can refer only to a lower planar dielectric. The default is the dielectric layer immediately below the conductor layer. Be aware that it is possible to create a conductor that cuts into a dielectric, which might not be the intended effect.

When used with a DIELECTRIC layer definition, the MEASURED_FROM keyword can refer to a lower dielectric or can have the value TOP_OF_CHIP.

The heights of the conductors and dielectrics are determined exclusively by the order in which they are specified and by the thicknesses of the lower layers. When you are specifying a new conductor or dielectric layer, the bottom plane of that layer is exactly the top plane of the dielectric layer immediately below it unless a MEASURED_FROM option is included (to explicitly specify the location of the bottom plane).

Examples

In the following example, TOP is planarized by measuring from D2:

```
DIELECTRIC TOP {
     THICKNESS = 3.6
     MEASURED_FROM = D2
```

```
     ER = 3.9
}
```

In the following example, D3 is a conformal dielectric:

```
DIELECTRIC D3 {
     THICKNESS=0.2
     MEASURED_FROM = TOP_OF_CHIP
     ER=3.9
}
```

See Also

- CONDUCTOR
- DIELECTRIC
- SW_T
- TW_T

Feedback

MEASURED_FROM_CONDUCTOR

Modifies the reference location where thickness is measured, starting from the top surface of a bottom conductor layer. Valid within a DIELECTRIC conformal block.

Syntax

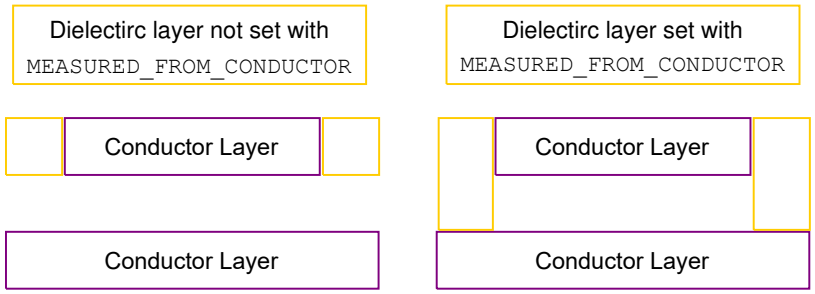
```
MEASURED_FROM_CONDUCTOR
```

Description

The MEASURED_FROM_CONDUCTOR option modifies the reference location where thickness is measured to account for process characteristics in conformal dielectrics layers.

When you use the DIELECTRIC layer (yellow block in Figure 259) definition by setting the MEASURED_FROM_CONDUCTOR keyword, this conformal dielectric layer starts measuring thickness from the top surface of the bottom conductor. The MEASURED_FROM_CONDUCTOR keyword can be used only in the dielectric layer with THICKNESS=0.

Figure 259 Dielectric Layer Set With MEASURED_FROM_CONDUCTOR



Examples

In the following example, TOP conformal layer is planarized (flattened) by measuring thickness from the bottom conductor:

```
DIELECTRIC TOP {
    IS_CONFORMAL
    MEASURED_FROM_CONDUCTOR
    ASSOCIATED_CONDUCTOR=M1
    THICKNESS=0
    SW_T=0.04
    TW_T=0.03
    BW_T=0.0032
    ER=4.9
}
```

Feedback

See Also

- CONDUCTOR
- DIELECTRIC
- SW_T
- TW_T
- BW_T

Feedback

MULTIGATE

Describes FinFET devices.

Syntax

```
MULTIGATE fin {
  FIN_SPACING = fin_spacing
  FIN_WIDTH = fin_width | FIN_WIDTHS {fw1 fw2 fw3... fwn}
  [FIN_LENGTH = fin_length]
  FIN_THICKNESS = fin_thickness
  RAISED_DIFFUSION_GROWTH = raised_diffusion_growth (optional)
  GATE_POLY_EXTENSION = gate_poly_extension (optional)
  GATE_OXIDE_TOP_T = gate_oxide_top_thickness
  GATE_OXIDE_SIDE_T = gate_oxide_side_thickness
  GATE_OXIDE_ER = gate_oxide_permittivity
  GATE_POLY_TOP_T = gate_poly_top_thickness
  GATE_POLY_SIDE_T = gate_poly_side_thickness
  CHANNEL_ER = channel_permittivity
  GATE_DIFFUSION_LAYER_PAIR { (PGATE PDIFF) (NGATE NDIFF) ...}
}
```

Arguments

Argument	Description
fin_spacing	Fin spacing; the distance between adjacent fin sidewalls. Ignored if multiple fin widths are supplied. Units: microns
fw1, fw2, ...	Fin widths. A separate capacitance model is generated for each specified fin width. Units: microns
fin_length	Fin length (optional); the distance between the gate sidewall and the raised diffusion region Units: microns
fin_thickness	Fin thickness Units: microns
raised_diffusion_growth	Raised diffusion growth (optional); distance that raised diffusion extends laterally from fin sidewall Units: microns
gate_poly_extension	Gate poly extension (optional); distance that gate poly extends laterally outside the diffusion region Units: microns

Feedback

Argument	Description
gate_oxide_top_thickness	Gate oxide top thickness Units: microns
gate_oxide_side_thickness	Gate oxide side thickness Units: microns Default: value of GATE_OXIDE_TOP_T
gate_oxide_permittivity	Gate oxide relative permittivity Units: none (ratio)
gate_poly_top_thickness	Gate poly top thickness Units: microns
gate_poly_side_thickness	Gate poly side thickness Units: microns Default: value of GATE_POLY_TOP_T
channel_permittivity	Channel relative permittivity Units: none (ratio)
GATE_DIFFUSION_LAYER_PAIR	Identifies the gate diffusion layer pair

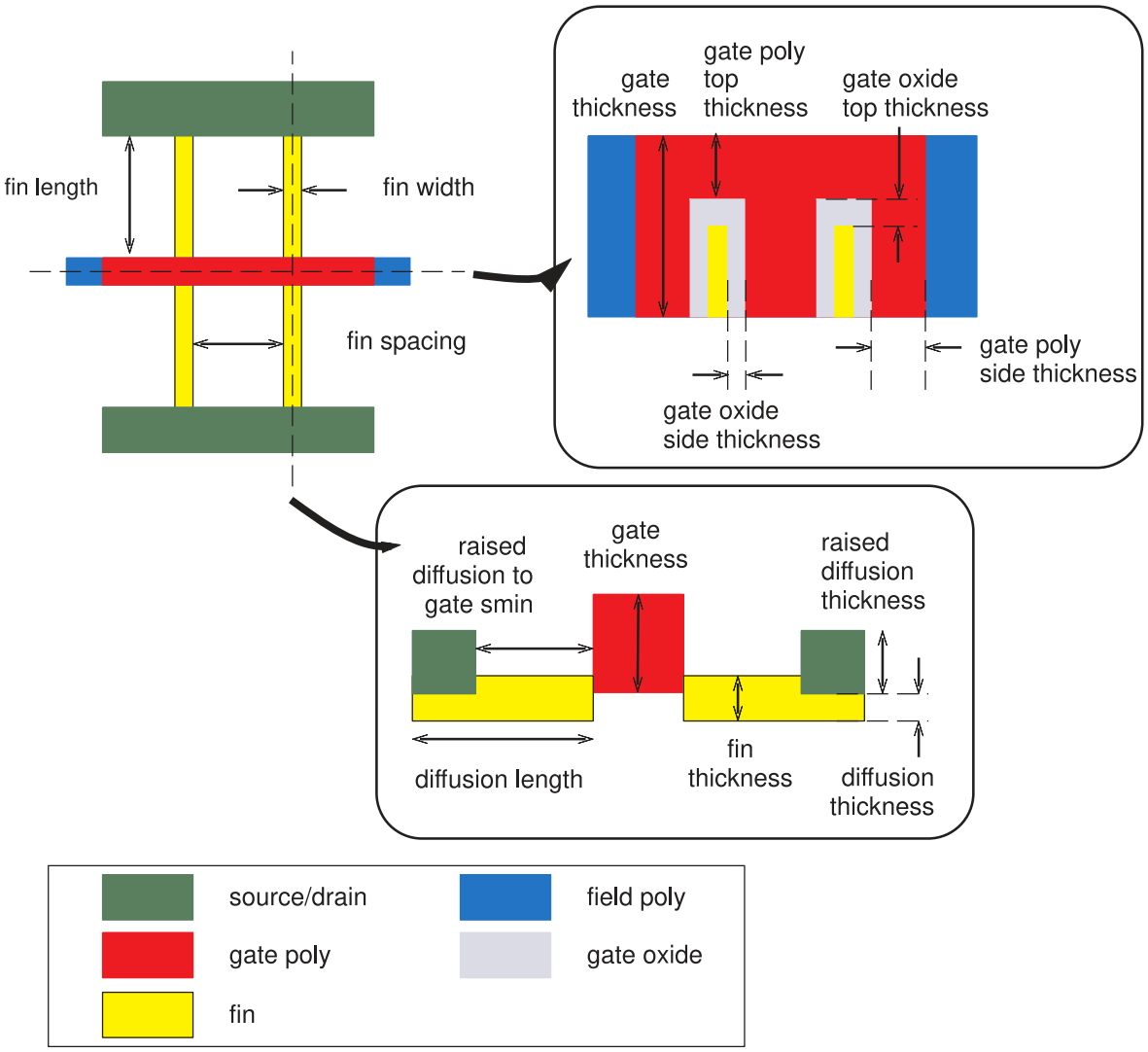
Description

The MULTIGATE option models FinFETs and other advanced devices. Inside a MULTIGATE block, you specify parameters that describe a three-dimensional structure. The units for all length parameters are microns. Figure 260 and Figure 261 illustrate FinFET parameters and cross sections.

The FIN_LENGTH parameter is optional. If the parameter is not specified, the final fin length is the distance from the gate edge to the far edge of the diffusion region, labeled as diffusion length in Figure 260.

Feedback

Figure 260 FinFET Top View and Cross Section Along Gate and Along Fin



The `RAISED_DIFFUSION_GROWTH` parameter is optional; it represents the distance that a raised source/drain region extends laterally from the fin sidewall. If the value is larger than half the spacing between the fins, the raised diffusion regions merge, as shown by the large green boxes in Figure 260. If the value is less than half the fin spacing, the raised diffusion regions remain separate, as shown in Figure 261.

Feedback

If the RAISED_DIFFUSION_GROWTH parameter is not specified, a merged region is assumed. If the parameter is specified, the dimensions of the unmerged raised diffusion box are as follows:

- Length (parallel to the fin length), calculated as:

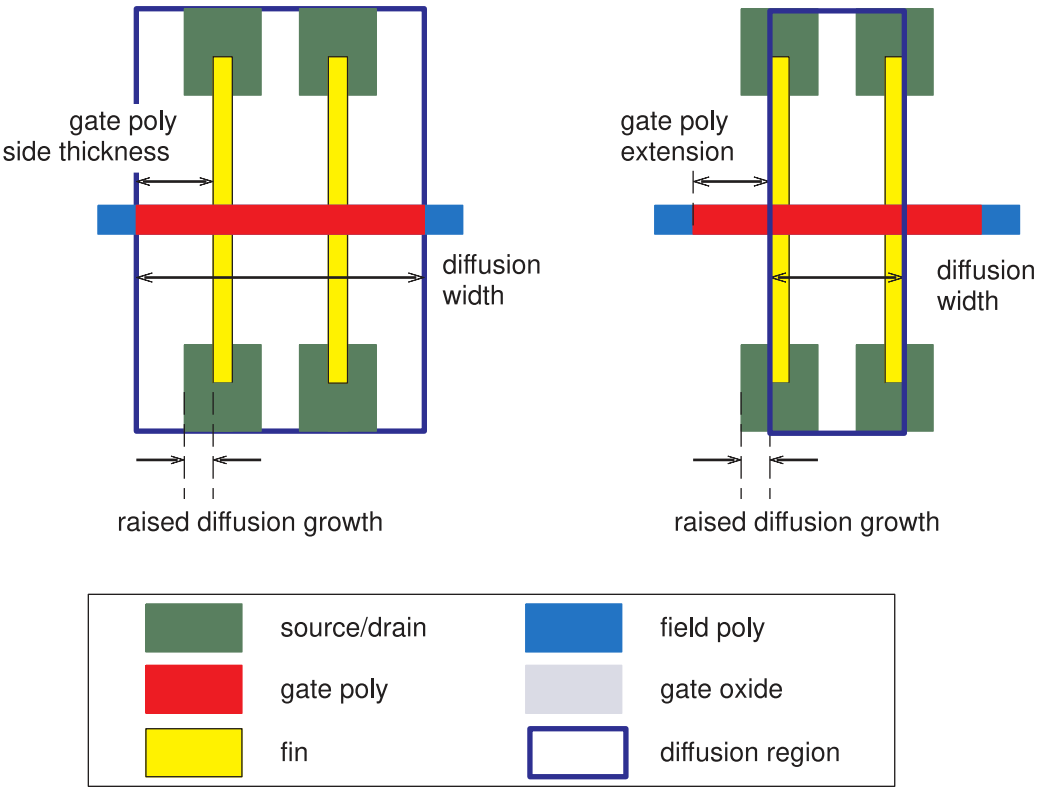
  DIFFUSION_LENGTH - RAISED_DIFFUSION_TO_GATE_SMIN

- Width (perpendicular to the fin length), calculated as:

  FIN_WIDTH + 2*RAISED_DIFFUSION_GROWTH

- Height or thickness, equal to the RAISED_DIFFUSION_THICKNESS parameter

Figure 261 FinFET Top View With Different Diffusion Bounding Boxes



Feedback

Two styles of diffusion regions are supported, as shown by the dashed lines in Figure 261.

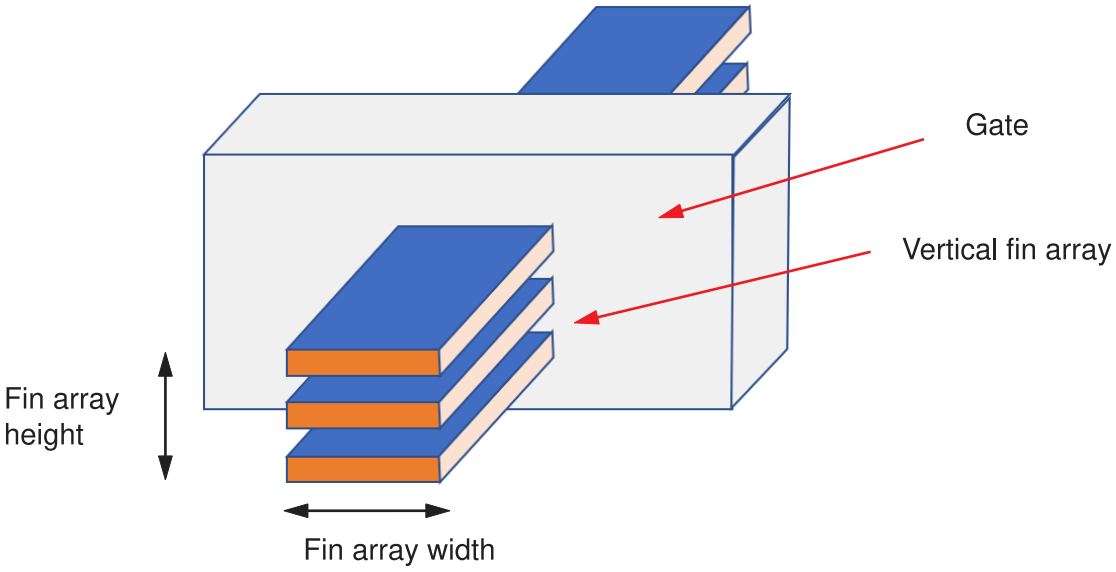
- If the GATE_POLY_SIDE_T parameter is specified and the GATE_POLY_EXTENSION parameter is undefined, the diffusion region boundary extends beyond the outermost fins by the distance specified in the GATE_POLY_SIDE_T parameter.
- If the GATE_POLY_EXTENSION parameter is defined, the diffusion region boundary coincides with the outer sidewall of the outermost fins; the gate poly extends beyond the diffusion region boundary by that distance.

Gate-All-Around Devices

Advanced transistors might have a gate-all-around structure, in which the gate polysilicon completely surrounds the fin. In addition, devices can have arrays of fins, as shown in Figure 262.

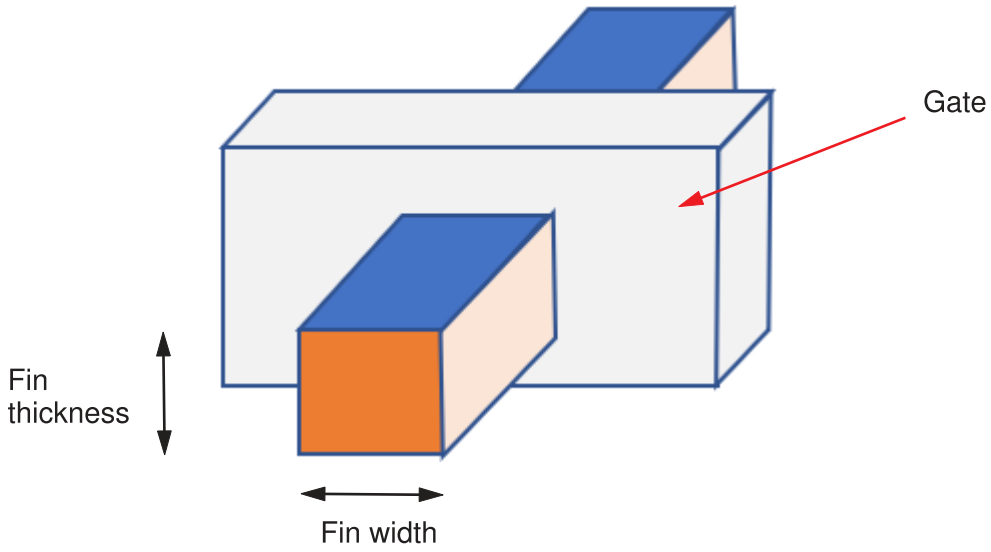
The foundry treats a vertical fin array as a single fin with an effective width and thickness to be compatible with the StarRC tool's FinFET modeling capabilities. Horizontal fin arrays are not supported.

Figure 262 Gate-All-Around FET With Vertical Fin Array



Feedback

Figure 263 Extraction Model of Vertical Fin Array



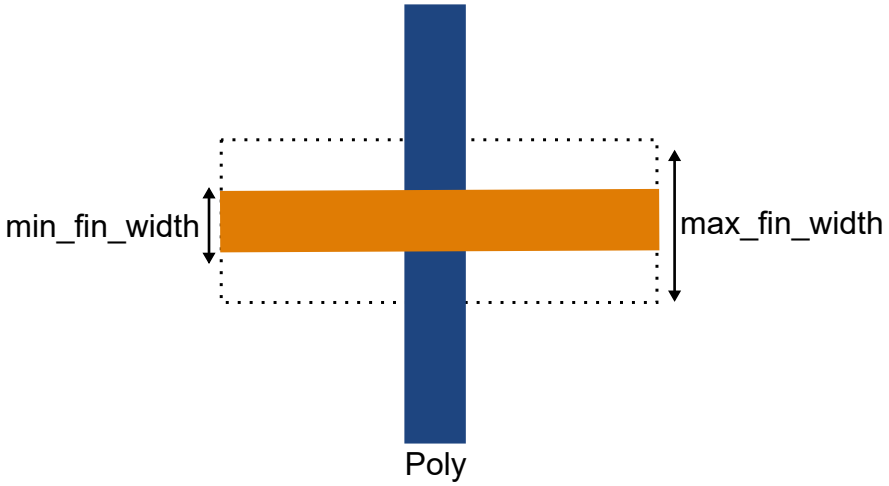
One-Fin FinFET Devices

You can use the *fw1, fw2, ...* arguments of the FIN_WIDTHS variable to support one-fin FinFET devices. Based on the number of values you specify with the FIN_WIDTHS keyword, the MULTIGATE command considers the maximum width as follows to model the variation range of the fin, see Figure 264:

- If you specify only two values, the first value is considered as the minimum width and the second value as the maximum width.
- If you specify more than two values, the first value is considered as the minimum width and the last value as the maximum width.

Feedback

Figure 264 Values Specified With FIN_WIDTHS for One-Fin FinFET Device




```
FIN_WIDTHS = {min_fin_width max_fin_width}
FIN_WIDTHS = {min_fin_width grid_val_1 grid_val_2 ... max_fin_width}
```

If you have used the FIN_WIDTHS keyword within a MULTIGATE statement, the StarRC tool does not allow the FIN_WIDTH or FIN_SPACING keyword in the same MULTIGATE statement. Therefore, you must specify a one-fin transistor on a separate MULTIGATE statement to have its own definition of GATE and DIFFUSION layers. To use multiple multigate devices, you can apply GATE_RESISTANCE_ADJUSTMENT_FACTOR, MOS_GATE_DELTA_RESISTANCE, and other commands for the gate features on all MULTIGATE devices, as shown in Example 44.

Specify Preferred Dimensions

If device-level distances for gate width, gate-to-contact distance, or trench contact width are constant or equal to a few fixed values in the layout, the distance can be specified in the MULTIGATE statement if

- The gate length is fixed in the layout

  Use GATE_WPREF to declare fixed gate width in the layout and the drawn values of the gate length within brackets {gw, gw2, ...}.

- The gate- to-contact minimum spacing is fixed in the layout

  Use GATE_TO_CONTACT_SPREF {*gtc*, *gtc2*, ...}.

- The trench contact width is fixed in the layout

  Use LOWER_CONTACT_WPREF {*tcw*, *tcw2*, ...}.

- The trench contact extension length is fixed in the layout

  Use TRENCH_CONTACT_EXTENSION_LPREF {*tcl1*, *tcl2*, ...}

Feedback

- The field poly extension length is fixed in the layout

  Use `FIELD_POLY_EXTENSION_LPREF {fpl1, fpl2, ...}`.

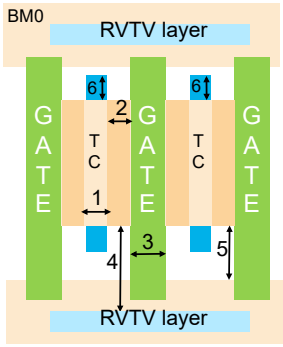
- There is no overhang between upper and lower trench contacts

  Use `STACKED_CONTACT_ALIGNMENT = SAME_WIDTH_ALIGNED`.

- The RVTV-to-diffusion layer preferred spacing is known

  Use `DIFFUSION_TO_RVTV_SPREF DISCRETE {rvtv1 rvtv2} {rvtv1, rvtv2}` when models are always computed with large RVTV and diffusion, the distance is a drawn layout parameter (that is, before applying any etch or expansion where).

Figure 265 Preferred Dimensions



1 LOWER_CONTACT_SPREF
2 GATE_TO_CONTACT_SPREF
3 GATE_WPREF
4 DIFFUSION_TO_RVTV_SPREF
5 FIELD_POLY_EXTENSION_LPREF
6 TRENCH_CONTACT_EXTENSION_LPREF

TC Trench contact layer

Examples

The following two `MULTIGATE` blocks model NMOS and PMOS FinFET transistors. In the presence of a `MULTIGATE` statement, the `DEVICE_TYPE` keyword must be used to identify the conductor layers that compose a specified device if there is more than one layer with `LAYER_TYPE=FIELD_POLY` in the ITF file.

Example 43 NMOS And PMOS FinFET Transistor Models

```
MULTIGATE Nmos_fin  {
                  Fin_spacing=0.040
                  Fin_width=0.02
                  Fin_length=0.06
                  Fin_thickness=0.05
                  Raised_diffusion_growth=0.01
                  Gate_oxide_top_t=0.003
                  Gate_oxide_side_t=0.003
                  Gate_oxide_er=8.0
                  Gate_poly_top_t=0.03
                  Gate_poly_side_t=0.04
                  Channel_er=10.0
                  Gate_diffusion_layer_pair {(Ngate  Ndiff) }
}
MULTIGATE Pmos_fin  {
```

```
                    Fin_spacing=0.045
                    Fin_width=0.025
                    Fin_length=0.065
                    Fin_thickness=0.05
                    Raised_diffusion_growth=0.01
                    Gate_oxide_top_t=0.0025
                    Gate_oxide_side_t=0.0025
                    Gate_oxide_er=8.0
                    Gate_poly_top_t=0.035
                    Gate_poly_side_t=0.045
                    Channel_er=10.0
                    Gate_diffusion_layer_pair {(Pgate  Pdiff) }
}
```

Example 44 One-Fin Transistor Model

```
MULTIGATE one_fin_lay {
FIN_WIDTHS={0.006 0.0012 0.0036 0.06}
FIN_LENGTH=0.08
FIN_THICKNESS=0.05
RAISED_DIFFUSION_GROWTH=0.01
GATE_OXIDE_TOP_T=0.003
GATE_OXIDE_SIDE_T=0.003
GATE_OXIDE_ER=8.0
GATE_POLY_TOP_T=0.03
GATE_POLY_SIDE_T=0.04
CHANNEL_ER=10.0
GATE_DIFFUSION_LAYER_PAIR {(Ngate Ndiff)
  }
```

If two device types have the same parameters, you can define them with a single `MULTIGATE` block as follows:

```
MULTIGATE MOS_FIN  {
          ...
          GATE_DIFFUSION_LAYER_PAIR { (NGATE NDIFF) (PGATE  PDIFF) }
}
```

See Also

- FinFET Modeling
- GATE_RESISTANCE_ADJUSTMENT_FACTOR
- MOS_GATE_DELTA_RESISTANCE
- RVTV Flow and Gate-All-Around FinFETs (GAAFETs) Extraction

Feedback

POLYNOMIAL_BASED_THICKNESS_VARIATION

Models conductor thickness variation as a function of feature density and width in a polynomial format. Valid within a CONDUCTOR block.

Syntax

```
POLYNOMIAL_BASED_THICKNESS_VARIATION {
     [SI_]DENSITY_POLYNOMIAL_ORDERS { do(n) do(n-1) … do(0) }
     [SI_]WIDTH_POLYNOMIAL_ORDERS { wo(m) wo(m-1) … wo(0) }
     WIDTH_RANGES { wt0 wt1 … }
     POLYNOMIAL_COEFFICIENTS {
        a(n,m)    a(n,m-1)   …  a(n,0)
        a(n-1,m)  a(n-1,m-1) …  a(n-1,0)
        …
        a(0,m)    a(0,m-1)   …  a(0,0)
     }
     POLYNOMIAL_COEFFICIENTS {
        b(n,m)    b(n,m-1)   …  b(n,0)
        b(n-1,m)  b(n-1,m-1) …  b(n-1,0)
        …
        b(0,m)    b(0,m-1)   …  b(0,0)
     }
     POLYNOMIAL_COEFFICIENTS {
        c(n,m)    c(n,m-1)   …  c(n,0)
        c(n-1,m)  c(n-1,m-1) …  c(n-1,0)
        …
        c(0,m)    c(0,m-1)   …  c(0,0)
     }
     ...
     [DENSITY_BOUNDS_VS_WIDTH {
        (wd1 dmin_wd1 dmax_wd1)
        (wd2 dmin_wd2  dmax_wd2)
        …
        (wdn dmin_wdn dmax_wdn)
     } ]
     [THICKNESS_BOUNDS {
        tmin tmax
     } ]
}
SYNTAX NOTES: If the WIDTH_RANGES keyword is not used, there must be
exactly one POLYNOMIAL_COEFFICIENTS table. If the WIDTH_RANGES keyword
is used, the number of POLYNOMIAL_COEFFICIENTS tables must be one more
than the number of widths in the WIDTH_RANGES argument list.
```

Arguments

Argument	Description
do(n) ... do(0)	Density exponents Format: integer

Feedback

Argument	Description
wo(n) ... wo(0)	Width exponents Format: integer
wt0, *wt1*, ...	Conductor widths that determine which coefficient table to use; must appear in ascending order Units: microns
a(n,m), *b(n,m)*, ...	Polynomial coefficients
wd1, *wd2*, ...	Conductor widths that define density bounds; must appear in ascending order Units: microns
dmin_wd1, *dmax_wd1*	Minimum and maximum density for the corresponding width
tmin	Minimum absolute thickness; must be smaller than conductor nominal thickness Units: microns
tmax	Maximum absolute thickness; must be larger than conductor nominal thickness Units: microns

Description

The POLYNOMIAL_BASED_THICKNESS_VARIATION command uses a polynomial to model conductor thickness variation as a function of density and width. You can use either the as-drawn or the post-etch values of density and width in the calculations.

The command contains one or more tables of polynomial coefficients. A specific table applies to a range of conductor widths defined by the WIDTH_RANGES keyword. The coefficients table is selected as follows:

- If the WIDTH_RANGES keyword is not used, one coefficients table is used for all conductor widths. Only one table should be provided.
- If the WIDTH_RANGES keyword has an argument list with one width value, there must be exactly two POLYNOMIAL_COEFFICIENTS tables. The first coefficients table applies to conductor widths less than or equal to the listed width, while the second table applies to conductor widths larger than the listed width.
- If the WIDTH_RANGES keyword has an argument list with more than one width value, the widths define a series of ranges. The first coefficients table applies to conductor widths less than or equal to the first width in the list. The second table applies to widths greater than the first width in the list and less than or equal to the second width, and so on. The last table applies to conductor widths larger than the last width in the list.

Feedback

Thickness variation is calculated as follows, where W is the conductor width and D is the density:

$$\frac{dT}{T} = \begin{bmatrix} D^{do(n)} & D^{do(n\text{-}1)} & \dots & D^{do(1)} & D^{do(0)} \end{bmatrix} \begin{bmatrix} a(n,m) & a(n,m\text{-}1) & \dots & a(n,1) & a(n,0) \\ a(n\text{-}1,m) & \dots & \dots & \dots & a(n\text{-}1,0) \\ \dots & \dots & \dots & \dots & \dots \\ a(1,m) & \dots & \dots & \dots & a(1,0) \\ a(0,m) & a(0,m\text{-}1) & \dots & a(0,1) & a(0,0) \end{bmatrix} \begin{bmatrix} W^{wo(m)} \\ W^{wo(m\text{-}1)} \\ \dots \\ W^{wo(1)} \\ W^{wo(0)} \end{bmatrix}$$

By default, feature density is calculated within a square box 50 microns on a side, centered on the conductor of interest. You can change the size of this box by using the `DENSITY_BOX_WEIGHTING_FACTOR` option.

The following example illustrates the calculation by using symbols to represent the coefficients. In this example, one coefficients table is used for all conductor widths because no `WIDTH_RANGES` keyword is included:

```
POLYNOMIAL_BASED_THICKNESS_VARIATION {
   DENSITY_POLYNOMIAL_ORDERS { 2 1 0 }
   WIDTH_POLYNOMIAL_ORDERS { 4 2 0 }
   POLYNOMIAL_COEFFICIENTS {
      a b c
      d e f
      g h i
   }
}
```

The resulting thickness variation equation is as follows:

$$\frac{dT}{T} = D^2 \cdot \left(aW^4 + bW^2 + cW^0\right) + D^1 \cdot \left(dW^4 + eW^2 + fW^0\right) + D^0 \cdot \left(gW^4 + hW^2 + iW^0\right)$$

The following sections explain specific applications of the `POLYNOMIAL_BASED_THICKNESS_VARIATION` command:

- Specifying As-Drawn or Silicon Values for Width and Density
- Density Bounds
- Thickness Bounds

Examples

In the following example, the first coefficients table is used for widths less than or equal to 0.27 microns, while the second table is used for widths greater than 0.27 microns. All widths and densities are calculated using post-etch (silicon) dimensions.

```
CONDUCTOR M1 { THICKNESS=0.18 SIDE_TANGENT = 0.0556
     POLYNOMIAL_BASED_THICKNESS_VARIATION {
        SI_DENSITY_POLYNOMIAL_ORDERS { 3 2 1 0 }
        SI_WIDTH_POLYNOMIAL_ORDERS { 4 3 2 1 0 }
```

```
            WIDTH_RANGES { 0.27 }
            $ Coefficients for width <= 0.27
            POLYNOMIAL_COEFFICIENTS {
               0 1.656E+03 -9.488E+02 1.731E+02 -1.041E+01
               0 -1.212E+03 6.935E+02 -1.262E+02 7.666E+00
               0 2.314E+02 -1.320E+02 2.400E+01 -1.580E+00
               0 -5.211E+00 3.417E+00 -6.853E-01 1.131E-01
            }
            $ Coefficients for width > 0.27
            POLYNOMIAL_COEFFICIENTS {
               1.027E-03 -2.006E-02 8.996E-02 -5.189E-02 -1.814E-01
               -2.805E-03 5.795E-02 -3.084E-01 4.211E-01 1.152E-01
               2.097E-03 -4.375E-02 2.394E-01 -3.662E-01 -2.697E-02
               -4.866E-04 1.001E-02 -5.416E-02 1.012E-01 4.308E-02
            }
            DENSITY_BOUNDS_VS_WIDTH {
               (0.1 0.05 0.97)
               (0.2 0.07 0.49)
               (0.3 0.10 0.52)
               (1.0 0.50 0.97)
            }
            THICKNESS_BOUNDS { 0.30 0.45 }
         }
}
```

Error Conditions

Several ITF commands provide alternative methods for modeling conductor thickness variation. Therefore, they cannot be specified for the same conducting layer when the `POLYNOMIAL_BASED_THICKNESS_VARIATION` command is used. These commands are as follows:

- `THICKNESS_VS_DENSITY`
- `THICKNESS_VS_DENSITY_AND_WIDTH`
- `THICKNESS_VS_WIDTH_AND_SPACING`

Specifying As-Drawn or Silicon Values for Width and Density

You can use either the as-drawn layout dimensions or post-etch (silicon) dimensions in the calculations, as follows:

- Conductor width
  - If you use the `WIDTH_POLYNOMIAL_ORDERS` keyword, conductor widths used in calculations are based on the as-drawn layout dimensions. This includes the widths used in the `WIDTH_RANGES` keyword and the widths (but not the densities) used in the `DENSITY_BOUNDS_VS_WIDTH` keyword.
  - If you use the `SI_WIDTH_POLYNOMIAL_ORDERS` keyword, conductor widths are based on the post-etch (silicon) dimensions.

Feedback

- Conductor density
  - If you use the DENSITY_POLYNOMIAL_ORDERS keyword, conductor densities used in calculations are based on the as-drawn layout dimensions. This includes the densities used in the DENSITY_BOUNDS_VS_WIDTH keyword.
  - If you use the SI_DENSITY_POLYNOMIAL_ORDERS keyword, conductor densities are based on the post-etch (silicon) dimensions.

Using the global USE_SI_DENSITY command in the ITF file might conflict with these keywords. Table 101 describes the allowed usages.

Table 101 Effect of USE_SI_DENSITY Command Settings

POLYNOMIAL_BASED_THICKNESS_VARIATION Keyword	USE_SI_DENSITY = YES	USE_SI_DENSITY = NO (or not used)
WIDTH_POLYNOMIAL_ORDERS	drawn width	drawn width
DENSITY_POLYNOMIAL_ORDERS	silicon density	drawn density
SI_WIDTH_POLYNOMIAL_ORDERS	error	silicon width
SI_DENSITY_POLYNOMIAL_ORDERS	error	silicon density

Density Bounds

Use the optional DENSITY_BOUNDS_VS_WIDTH keyword to limit the conductor densities to be used in the calculations. Each entry in the argument list consists of one width value and two density values enclosed in parentheses; you can provide more than one such entry.

Whether the width and density values in the DENSITY_BOUNDS_VS_WIDTH keyword use as-drawn or post-etch (silicon) values is determined by the presence or absence of the SI_ prefix in the WIDTH_POLYNOMIAL_ORDERS and DENSITY_POLYNOMIAL_ORDERS keywords.

The density bounds in the DENSITY_BOUNDS_VS_WIDTH keyword are applied as follows:

- If the density is outside the range defined by the minimum and maximum densities, the nearest density bound is used in the calculation.
- If there is only one entry (in other words, one triplet of values) in the DENSITY_BOUNDS_VS_WIDTH argument list, the minimum and maximum densities apply to all conductor widths regardless of the width specified in the triplet.

- If there are multiple triples in the keyword argument list, the behavior is as follows:
  - If the conductor width is smaller than the smallest width specified, the minimum and maximum densities for the smallest width are used.
  - If the conductor width is larger than the largest width specified, the minimum and maximum densities for the largest width are used.
  - If the conductor width is between two specified widths, the minimum and maximum densities are determined by interpolating between the enclosing density values.

In the following example, only density values between 0.05 and 1.00 are used when calculating thickness variation for conductor widths up to and including 0.1 microns. Density values from 0.07 to 0.50 are used for conductor widths greater than 0.2 microns. If the conductor width is 0.15 microns, the allowable range of density values is 0.06 to 0.75, as determined by linear interpolation.

```
DENSITY_BOUNDS_VS_WIDTH {
   (0.1 0.05 1.00)
   (0.2 0.07 0.50)
}
```

If you do not use the `DENSITY_BOUNDS_VS_WIDTH` keyword, the density bounds are calculated using the minimum and maximum width and spacing values in the `ETCH_VS_WIDTH_AND_SPACING` command defined for the same conductor. This method is available for backward compatibility, but might result in unreasonable values.

In this method, the minimum and maximum densities are calculated as follows, where D is density, W is width, and S is spacing:

$$D_{min} = \frac{W_{min}}{(W_{min} + S_{max})}$$

$$D_{max} = \frac{W_{max}}{(W_{max} + S_{min})}$$

Thickness Bounds

Use the optional `THICKNESS_BOUNDS` keyword to apply limits to the calculated conductor thickness values, as follows:

- If the conductor thickness calculated from the polynomial function is smaller than the minimum thickness in the `THICKNESS_BOUNDS` argument list, the minimum thickness value is used instead of the calculated thickness.
- If the conductor thickness calculated from the polynomial function is larger than the maximum thickness in the `THICKNESS_BOUNDS` argument list, the maximum thickness value is used instead of the calculated thickness.

Feedback

See Also

- THICKNESS_VS_DENSITY
- THICKNESS_VS_DENSITY_AND_WIDTH
- THICKNESS_VS_WIDTH_AND_SPACING
- DENSITY_BOX_WEIGHTING_FACTOR
- ETCH_VS_WIDTH_AND_SPACING

Feedback

PROCESS_CORNER

Specifies an optional foundry identifier as part of a process information keyword group.

Syntax

```
PROCESS_CORNER  = corner_id
```

Arguments

Argument	Description
corner_id	One-word string, intended to identify the process corner

Description

The PROCESS_CORNER keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS_CORNER keyword are as follows:

- The keyword is a one-word string; spaces are not allowed.
- Case does not matter; TYPICAL is equivalent to typical.
- The special characters @, #, and $ are not allowed.

Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

Feedback

PROCESS_FOUNDRY

Specifies an optional foundry identifier as part of a process information keyword group.

Syntax

```
PROCESS_FOUNDRY = foundry_id
```

Arguments

Argument	Description
foundry_id	One-word string, intended to identify the foundry

Description

The PROCESS_FOUNDRY keyword is one of the five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS_FOUNDRY keyword are as follows:

- This keyword is a one-word string; spaces are not allowed.
- Case does not matter; FAB_1802c is equivalent to fab_1802C.
- The special characters @, #, and $ are not allowed.

Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

Feedback

PROCESS_NODE

Specifies an optional foundry identifier as part of a process information keyword group.

Syntax

```
PROCESS_NODE  = node_number
```

Arguments

Argument	Description
node_number	Floating-point number, intended to identify the technology node

Description

The PROCESS_NODE keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS_NODE keyword are as follows:

- The keyword is a nonnegative floating-point number.
- The value is intended to represent a technology node, but there are no requirements for the value.

Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

Feedback

PROCESS_TYPE

Specifies an optional foundry identifier as part of a process information keyword group.

Syntax

```
PROCESS_TYPE  = type_id
```

Arguments

Argument	Description
type_id	One-word string, intended to identify the process

Description

The PROCESS_TYPE keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS_TYPE keyword are as follows:

- This keyword is a one-word string; spaces are not allowed.
- Case does not matter; SOI is equivalent to soi.
- The special characters @, #, and $ are not allowed.

Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

Feedback

PROCESS_VERSION

Specifies an optional foundry identifier as part of a process information keyword group.

Syntax

```
PROCESS_VERSION  = version_number
```

Arguments

Argument	Description
version_number	Floating-point number, intended to identify the process version

Description

The PROCESS_VERSION keyword is one of five optional keywords that allow you to provide process-specific information at the beginning of an ITF file.

If you use the process description keywords, you must

- Specify all five keywords
- Place the keywords immediately after the TECHNOLOGY statement
- Specify the keywords in the correct order

Rules for the PROCESS_VERSION keyword are as follows:

- The keyword is a nonnegative floating-point number.
- The value is intended to represent a process version such as 1.0 or 2.0, but there are no requirements for the value.

Examples

The following example shows the five process information keywords in the required order:

```
TECHNOLOGY = abc
PROCESS_FOUNDRY = fab_1802C
PROCESS_NODE = 130
PROCESS_TYPE = SOI
PROCESS_VERSION = 2.0
PROCESS_CORNER = TYPICAL
```

Feedback

RAISED_DIFFUSION_ETCH

Specifies the etch distance of the raised diffusion conductor. Valid within a `CONDUCTOR` block.

Syntax

```
RAISED_DIFFUSION_ETCH = distance
```

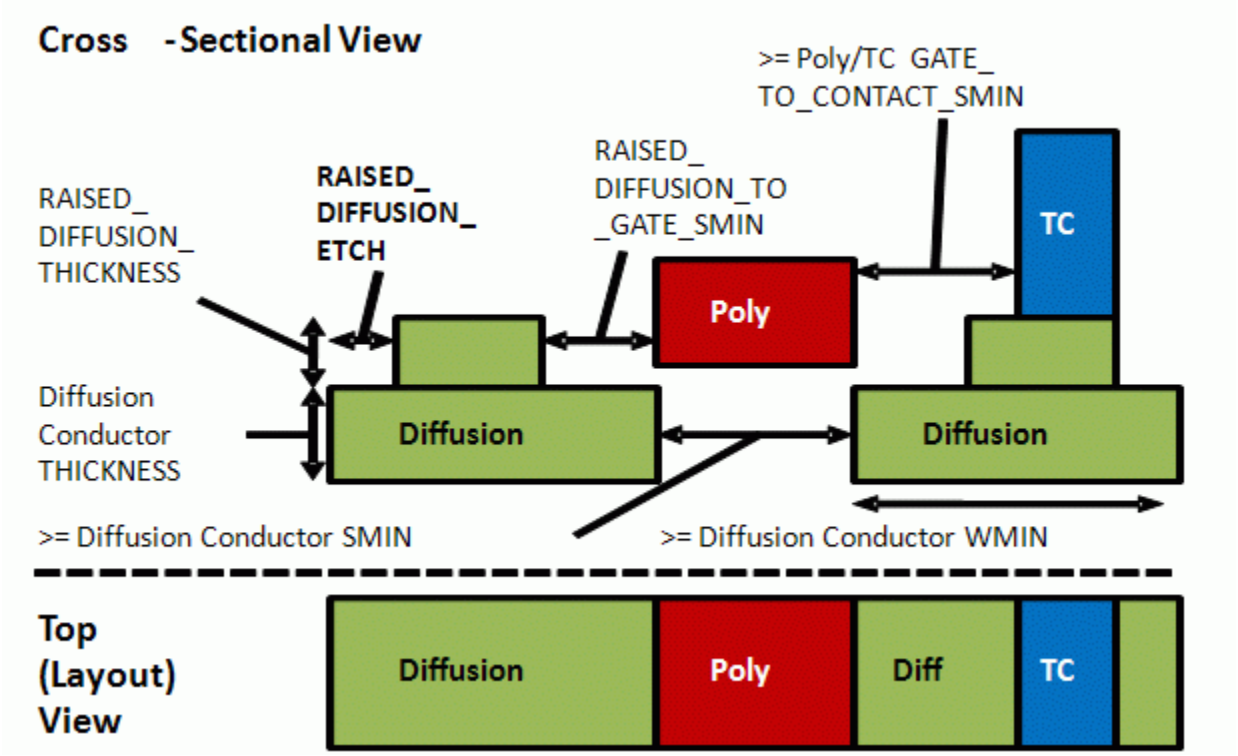
Arguments

Argument	Description
distance	Etch distance of the raised diffusion conductor; must be positive Units: microns

Description

The `RAISED_DIFFUSION_ETCH` option specifies the etch distance of the raised diffusion conductor, on the sides of the diffusion conductor that are not adjacent to a gate or field polysilicon conductor, as shown in Figure 266.

Figure 266 Cross Sectional and Top Views of the Trench Contact Process



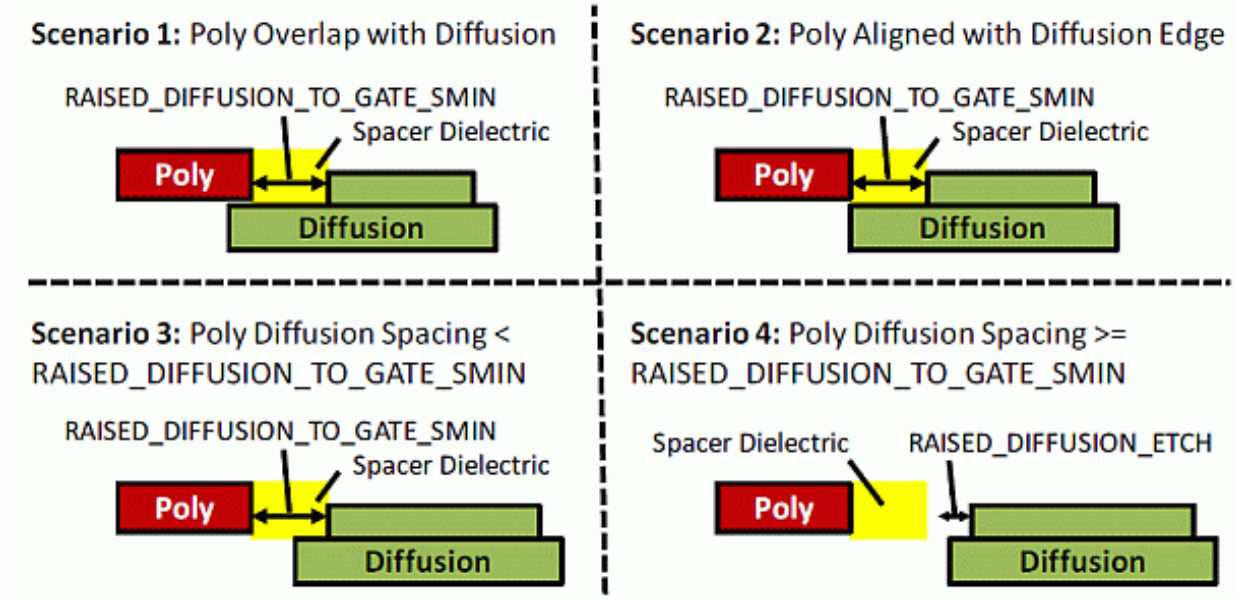
Feedback

Figure 267 shows the specific scenarios in which the RAISED_DIFFUSION_ETCH and RAISED_DIFFUSION_TO_GATE_SMIN options are applied. In a typical process, raised diffusion growth is affected by the location of the polysilicon spacer dielectric. The RAISED_DIFFUSION_TO_GATE_SMIN option models this process effect.

The RAISED_DIFFUSION_ETCH option models a different process effect in regions that do not overlap with the polysilicon spacer dielectric. The raised diffusion edge geometry is solely determined by the RAISED_DIFFUSION_THICKNESS, RAISED_DIFFUSION_TO_GATE_SMIN, and RAISED_DIFFUSION_ETCH options. Therefore, the RAISED_DIFFUSION_TO_GATE_SMIN option is independent of the polysilicon conductor's conformal dielectrics.

The RAISED_DIFFUSION_ETCH and RAISED_DIFFUSION_TO_GATE_SMIN options are applied based on the post-etch silicon dimensions of the polysilicon and diffusion conductors.

Figure 267 Polysilicon Spacing From Raised Diffusion Regions



The RAISED_DIFFUSION_ETCH option has the following constraints:

- The RAISED_DIFFUSION_ETCH option must be specified with both the RAISED_DIFFUSION_THICKNESS and RAISED_DIFFUSION_TO_GATE_SMIN options.
- The RAISED_DIFFUSION_ETCH option must be specified in a conductor layer that contains the LAYER_TYPE = DIFFUSION specification.
- The value of the RAISED_DIFFUSION_ETCH option must not generate nonphysical conductor geometries.

If any of these constraints are violated in a given ITF file, the grdgenxo tool issues an error message.

Examples

```
CONDUCTOR DIFF {
  THICKNESS = 0.05
  LAYER_TYPE = DIFFUSION
  RAISED_DIFFUSION_ETCH = 0.005
  RAISED_DIFFUSION_THICKNESS = 0.015
  RAISED_DIFFUSION_TO_GATE_SMIN = 0.01
  …
}
```

See Also

- CONDUCTOR
- LAYER_TYPE
- RAISED_DIFFUSION_THICKNESS
- RAISED_DIFFUSION_TO_GATE_SMIN

Feedback

RAISED_DIFFUSION_ETCH_TABLE

Specifies the device-dependent etch distance of the raised diffusion conductor. Valid within a CONDUCTOR block.

Syntax

```
RAISED_DIFFUSION_ETCH_TABLE {
  (device_type1 distance1)
  (device_type2 distance2)
  …
}
```

Arguments

Argument	Description
device_type	Device type
distance	Etch distance of the raised diffusion conductor Units: microns

Description

The RAISED_DIFFUSION_ETCH_TABLE option specifies the device-dependent etch distance of the raised diffusion conductor on the sides of the diffusion conductor that are not adjacent to a gate conductor. The RAISED_DIFFUSION_ETCH keyword specifies the default that is used if the device type is not found in the table.

The device type of the raised source and drain is determined by the surrounding gate device type. The spacing corresponding to the resulting device type is used for the source and drain.

See Also

- CONDUCTOR
- DEVICE_TYPE
- LAYER_TYPE
- RAISED_DIFFUSION_THICKNESS
- RAISED_DIFFUSION_TO_GATE_SMIN_TABLE

Feedback

RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER

Specifies the dielectric constant of the area between the raised diffusion and the gate processes. Valid within a CONDUCTOR block.

Syntax

```
RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER = er_value
```

Arguments

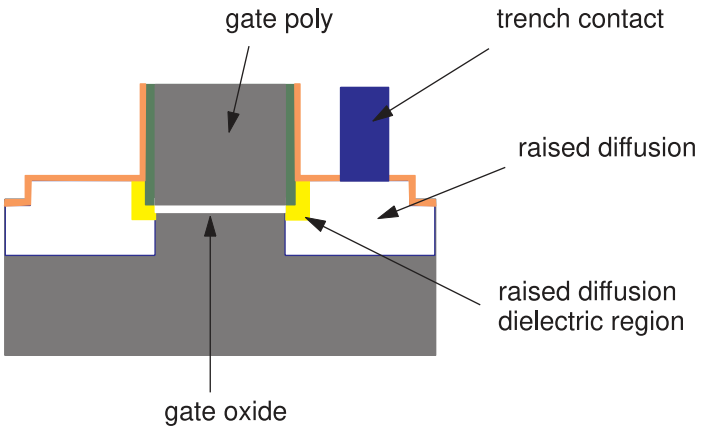
Argument	Description
er_value	The dielectric constant of the area between the raised diffusion and the gate processes Units: microns Default: Dielectric constant of the raised diffusion area.

Description

The RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER option specifies the dielectric constant of the dielectric region between the raised diffusion and the gate processes when you model silicon dielectrics underneath the gate and diffusion conductors, as shown in Figure 268. This option works with the BW_T option, which models the associated silicon dielectrics.

Place the RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER option within the raised diffusion CONDUCTOR statement. This option can only be placed in conductors containing RAISED_DIFFUSION_THICKNESS and RAISED_DIFFUSION_TO_GATE_SMIN definitions.

Figure 268 Area Between the Raised Diffusion and the Gate Processes



Examples

The following example uses the `RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER` option to specify the dielectric constant of the raised diffusion dielectric region, shown in Figure 268.

```
CONDUCTOR DIFF { THICKNESS=0.1
RAISED_DIFFUSION_THICKNESS=0.02
RAISED_DIFFUSION_TO_GATE_SMIN=0.03
RAISED_DIFFUSION_GATE_SIDE_CONFORMAL_ER=5.0 … }
```

See Also

- MEASURED_FROM
- SW_T
- BW_T
- THICKNESS

Feedback

RAISED_DIFFUSION_THICKNESS

Specifies additional diffusion thickness in raised source and drain regions. Valid within a CONDUCTOR block.

Syntax

```
RAISED_DIFFUSION_THICKNESS = thickness
```

Arguments

Argument	Description
thickness	Additional diffusion thickness in raised source and drain regions Units: microns

Description

The RAISED_DIFFUSION_THICKNESS option specifies additional diffusion thickness in raised source and drain regions, as shown in Figure 266.

Specify the RAISED_DIFFUSION_THICKNESS option within a CONDUCTOR statement with LAYER_TYPE = DIFFUSION. The raised portion of the diffusion is not applied to regions that are closer than the specified RAISED_DIFFUSION_TO_GATE_SMIN value to conductors with the GATE or FIELD_POLY layer type.

Examples

In the following example, the raised diffusion region exists in all diffusion conductors at a spacing of 10 nm from adjacent polysilicon conductors. The raised diffusion region extends 11 nm above the nominal diffusion height. Therefore, the raised diffusion region is covertical with the bottom 10 nm of the polysilicon conductor. The DIFF_NO_RSD layer is a standard diffusion layer without raised source and drain regions, which can also exist in the process.

Example 45 Raised Diffusion Definition

```
CONDUCTOR PS {
  THICKNESS=0.04
  WMIN=0.04
  SMIN=0.04
  GATE_TO_CONTACT_SMIN=0.02
  LAYER_TYPE=GATE
  …
}
DIELECTRIC DP1 {
  THICKNESS=0.001
  …
}
```

Feedback

```
DIELECTRIC D_DIFF {
  THICKNESS=0.04
  …
}
CONDUCTOR DIFF {
  THICKNESS=0.04
  WMIN=0.04
  SMIN=0.04
  RAISED_DIFFUSION_THICKNESS=0.011
  RAISED_DIFFUSION_TO_GATE_SMIN=0.01
  LAYER_TYPE=DIFFUSION
  …
}
CONDUCTOR DIFF_NO_RSD {
  THICKNESS=0.04
  WMIN=0.04
  SMIN=0.04
  LAYER_TYPE=DIFFUSION
  …
}
```

See Also

- CONDUCTOR
- LAYER_TYPE
- RAISED_DIFFUSION_ETCH
- RAISED_DIFFUSION_TO_GATE_SMIN

Feedback

RAISED_DIFFUSION_TO_GATE_SMIN

Specifies the minimum lateral spacing between raised source and drain regions and gate or field polysilicon conductors. Valid within a `CONDUCTOR` block.

Syntax

```
RAISED_DIFFUSION_TO_GATE_SMIN = spacing
```

Arguments

Argument	Description
spacing	Minimum lateral spacing between raised source and drain regions Units: microns

Description

The `RAISED_DIFFUSION_TO_GATE_SMIN` option specifies the minimum lateral spacing between raised source and drain regions and gate or field polysilicon conductors, as shown in Figure 266.

Specify the `RAISED_DIFFUSION_TO_GATE_SMIN` option within a `CONDUCTOR` block that contains a `LAYER_TYPE = DIFFUSION` statement.

See Also

- CONDUCTOR
- LAYER_TYPE
- RAISED_DIFFUSION_ETCH
- RAISED_DIFFUSION_THICKNESS

Feedback

RAISED_DIFFUSION_TO_GATE_SMIN_TABLE

Specifies the device-dependent minimum lateral spacing between raised source and drain regions and gate conductors. Valid within a CONDUCTOR block.

Syntax

```
RAISED_DIFFUSION_TO_GATE_SMIN_TABLE {
  (device_type1 spacing1)
  (device_type2 spacing2)
  …
}
```

Arguments

Argument	Description
device_type	Device type
spacing	Minimum lateral spacing between raised source and drain regions Units: microns

Description

The RAISED_DIFFUSION_TO_GATE_SMIN_TABLE option specifies the device-dependent minimum lateral spacing between raised source and drain regions and gate conductors. The RAISED_DIFFUSION_TO_GATE_SMIN keyword specifies the default that is used if the device type is not found in the table.

Specify the RAISED_DIFFUSION_TO_GATE_SMIN_TABLE option within a CONDUCTOR statement that contains the LAYER_TYPE = DIFFUSION statement.

The device type of the raised source and drain is determined by the surrounding gate device type. The spacing corresponding to that device type is used for the source and drain.

Examples

```
RAISED_DIFFUSION_TO_GATE_SMIN_TABLE
  {(G_1D5VIO_PMOS 0.02) (G_CORE_PMOS 0.015)}
```

See Also

- CONDUCTOR
- LAYER_TYPE
- RAISED_DIFFUSION_TO_GATE_SMIN

Feedback

REFERENCE_DIRECTION

Specifies the reference direction of the process technology.

Syntax

```
REFERENCE_DIRECTION = VERTICAL | HORIZONTAL | GATE
```

Arguments

Argument	Description
VERTICAL	Reference direction is vertical
HORIZONTAL	Reference direction is horizontal
GATE	Reference direction is determined by device pin layout

Description

The REFERENCE_DIRECTION statement defines the reference direction for the application of orientation-dependent etch defined by the ETCH_VS_WIDTH_AND_SPACING statement.

The REFERENCE_DIRECTION statement appears in the header of an ITF file with other global statements such as the TECHNOLOGY statement. If the ITF file does not contain a reference direction specification, the StarRC tool issues an error message and exits.

If the reference direction is also specified in the StarRC command file, the setting in the command file takes precedence.

If you set the reference direction to GATE, the StarRC tool uses the locations of the gate, drain, and source pins of the first encountered transistor to determine whether the reference direction should be VERTICAL or HORIZONTAL. The same reference direction is used for all devices in the design.

Examples

The following example specifies that the reference direction is horizontal:

```
REFERENCE_DIRECTION = HORIZONTAL
```

See Also

- ETCH_VS_WIDTH_AND_SPACING
- REFERENCE_DIRECTION (StarRC command)

Feedback

RESISTIVE_ONLY_ETCH

Identical to the ETCH option, except that only resistance is affected. Valid within a CONDUCTOR block.

Syntax

```
RESISTIVE_ONLY_ETCH = etch_value
```

Arguments

Argument	Description
etch_value	Absolute width adjustment for one sidewall. A positive value shrinks the conductor; a negative value expands it. Units: microns

Description

The RESISTIVE_ONLY_ETCH option applies an etch value to the sidewalls of a conductor. A positive value denotes conductor shrink; a negative value denotes conductor expansion. The adjusted conductor width is equal to the drawn width minus twice the etch value.

Use this option instead of the ETCH option to specify that an etch operation is to be used only for resistance calculations.

If you use one of the ETCH options in addition to one or more ETCH_VS_WIDTH_AND_SPACING tables, the ETCH_VS_WIDTH_AND_SPACING operations are applied first, followed by the ETCH operation.

This option is not the same as ETCH_VS_WIDTH_AND_SPACING RESISTIVE_ONLY.

This option works only with EXTRACTION:RC. Otherwise resistance extraction does not have etching effects.

Examples

```
CONDUCTOR metal1 {
     RESISTIVE_ONLY_ETCH = 0.05
     THICKNESS=0.66
     WMIN=0.15 SMIN=0.15 RPSQ=0.078
}
```

See Also

- CAPACITIVE_ONLY_ETCH
- ETCH

Feedback

RHO

Defines the bulk resistivity of a via or conductor layer.

Syntax

```
RHO = rho_value
```

Arguments

Argument	Description
rho_value	Bulk resistivity of the via or conductor layer Units: ohm-microns Default for conductors: 0.0 Default for vias: RPV x AREA equal to 1.0e-6

Description

The RHO option specifies bulk resistivity and can be used for either via or conductor layers.

In both cases, multiple methods are available for specifying layer resistance properties. See the reference pages for the CONDUCTOR and VIA options for more information about alternative methods.

Errors

The following options are mutually exclusive methods for specifying conductor resistance: RPSQ, RPSQ_VS_WIDTH_AND_SPACING, RPSQ_VS_SI_WIDTH, RHO, RHO_VS_SI_WIDTH_AND_THICKNESS, and RHO_VS_WIDTH_AND_SPACING.

Examples

```
VIA via1 {FROM=M1 TO=M2 RHO=0.263}
CONDUCTOR M1 {THICKNESS=0.4 SMIN=0.15 WMIN=0.18 RHO=0.8}
```

See Also

- CONDUCTOR
- VIA

Feedback

RHO_VS_SI_WIDTH_AND_THICKNESS

Models resistivity as a function of silicon (post-etch) width and thickness. Valid within a CONDUCTOR block.

Syntax

```
RHO_VS_SI_WIDTH_AND_THICKNESS {
  WIDTH { w1 w2 ... wn}
  THICKNESS { t1 t2 ... tm}
  VALUES { v(w1,t1) v(w2,t1) ... v(wn,t1)
           v(w1,t2) v(w2,t2) ... v(wn,t2)
           …
           v(w1,tm) v(w2,tm) ... v(wn,tm)    }
}
```

Arguments

Argument	Description
w1 w2 w3 …	Silicon widths of the conductor Units: microns
t1 t2 t3 …	Silicon thicknesses of the conductor Units: microns
v(w1,t1) v(w2,t1) …	Resistivity values for the corresponding width and thickness Units: ohm-microns The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Description

The RHO_VS_SI_WIDTH_AND_THICKNESS option models resistivity as a function of silicon width and thickness. Silicon width and thickness are post-etch values.

Feedback

The following usage notes apply:

- You must specify one of the following ITF options for the conductor:
  - POLYNOMIAL_BASED_THICKNESS_VARIATION
  - THICKNESS_VS_DENSITY
  - THICKNESS_VS_WIDTH_AND_SPACING
- You must specify one of the following ITF options for the conductor:
  - ETCH
  - ETCH_VS_WIDTH_AND_SPACING (RESISTIVE_ONLY)

Errors

The following options are mutually exclusive methods for specifying conductor resistance: RPSQ, RPSQ_VS_WIDTH_AND_SPACING, RPSQ_VS_SI_WIDTH, RHO, RHO_VS_SI_WIDTH_AND_THICKNESS, and RHO_VS_WIDTH_AND_SPACING.

Examples

```
RHO_VS_SI_WIDTH_AND_THICKNESS {
     WIDTH {0.1 0.2 0.3 0.4}
     THICKNESS {0.11 0.22 0.33}
     VALUES { 0.304 0.410 0.518 0.640
              0.210 0.340 0.438 0.560
              0.504 0.530 0.618 0.720
     }
}
```

See Also

- POLYNOMIAL_BASED_THICKNESS_VARIATION
- THICKNESS_VS_DENSITY
- THICKNESS_VS_WIDTH_AND_SPACING
- ETCH
- ETCH_VS_WIDTH_AND_SPACING

Feedback

RHO_VS_WIDTH_AND_SPACING

Models resistivity variation with respect to width and spacing. Valid within a CONDUCTOR block.

Syntax

```
RHO_VS_WIDTH_AND_SPACING {
     SPACINGS {s1 s2 ... sn}
     WIDTHS   {w1 w2 ... wm}
     VALUES   {v(s1,w1) v(s2,w1) ... v(sn,w1)
               v(s1,w2) v(s2,w2) ... v(sn,w2)
               ...
               v(s1,wm) v(s2,wm) ... v(sn,wm)  }
     }
```

Arguments

Argument	Description
s1 s2 ...	Spacings to the nearest conductor Units: microns
w1 w2 ...	Widths of the nearest conductor Units: microns
v(s1,w1) ...	Resistivity values for the corresponding width and spacing Units: ohm-microns The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Description

Specify this option to model conductor resistivity variation with respect to width and spacing.

The following usage notes apply:

- If the width or spacing falls between two indexes, the tool performs linear interpolation to calculate the RPSQ value.
- If the width or spacing falls outside the table boundaries, the tool uses the nearest boundary to calculate the RPSQ value.

Errors

The following options are mutually exclusive methods for specifying conductor resistance: RPSQ, RPSQ_VS_WIDTH_AND_SPACING, RPSQ_VS_SI_WIDTH, RHO, RHO_VS_SI_WIDTH_AND_THICKNESS, and RHO_VS_WIDTH_AND_SPACING.

Feedback

See Also

- RHO
- RHO_VS_SI_WIDTH_AND_THICKNESS

Feedback

RPSQ

Specifies the resistance per square (RPSQ) of a conductor layer, based on the silicon width (physical width). Valid within a CONDUCTOR block.

Syntax

```
RPSQ = rpsq_value
```

Arguments

Argument	Description
rpsq_value	Sheet resistance of the conducting layer Default: 0 Units: ohms/square

Description

Resistance per square (RPSQ) is the sheet resistance of a conductor, based on the silicon width (physical width), not the drawn width. You can specify the resistive properties of conductor layers using either RPSQ or rho (resistivity) values, but only one is required.

Note:

You can also specify the RPSQ value in the mapping file within the conducting_layers statement. The RPSQ value specified in the mapping file overrides any RPSQ, RPSQ_VS_WIDTH_AND_SPACING, or RPSQ_VS_SI_WIDTH statements in the ITF file.

The RPSQ value is also modified by the conductor thickness variation specified by the BOTTOM_THICKNESS_VS_SI_WIDTH statement, unless it contains the CAPACITIVE_ONLY keyword.

Errors

The following options are mutually exclusive methods for specifying conductor resistance: RPSQ, RPSQ_VS_WIDTH_AND_SPACING, RPSQ_VS_SI_WIDTH, RHO, RHO_VS_SI_WIDTH_AND_THICKNESS, and RHO_VS_WIDTH_AND_SPACING.

Examples

```
CONDUCTOR metal1 {
     RESISTIVE_ONLY_ETCH=0.05 THICKNESS=0.66
     WMIN=0.15 SMIN=0.15 RPSQ=0.078
}
```

Feedback

See Also

- RHO
- RPSQ_VS_SI_WIDTH
- RPSQ_VS_WIDTH_AND_SPACING

Feedback

RPSQ_VS_SI_WIDTH

Defines the conductor sheet resistance as a function of the conductor silicon width (physical width). Valid in a CONDUCTOR definition.

Syntax

```
RPSQ_VS_SI_WIDTH {
      [NUMBER_OF_TABLES = num_of_tables]
      [table_name1] {
            (w1, r1)
            (w2, r2)
            ...
            (wn, rn)
      }
      [table_name2 {
            (w1, r1)
            (w2, r2)
            ...
            (wn, rn)
      }]
      ...
}
```

Arguments

Argument	Description
num_of_tables	Number of tables defined for this conductor; an integer greater than or equal to 1
table_name1, *table_name2*, ...	Table names
w1, w2, ...	Post-etch conductor width Units: microns
r1, r2, ...	Conductor sheet resistance for the corresponding width Units: ohms/square

Description

The RPSQ_VS_SI_WIDTH table specifies conductor sheet resistance (resistance per square, or RPSQ) as a function of conductor width. Resistance might vary with line width due to effects such as cladding and dishing.

The conductor widths in this table are post-etch widths, also known as silicon width or physical width. The RPSQ values are the sheet resistance values that correspond to the silicon widths, not to the drawn widths.

Feedback

Note:

You can also specify the RPSQ value in the mapping file within the `conducting_layers` statement. The RPSQ value specified in the mapping file overrides all `RPSQ`, `RPSQ_VS_WIDTH_AND_SPACING`, or `RPSQ_VS_SI_WIDTH` tables in the ITF file.

The resistance of a conductor layer is also modified by the conductor thickness variation specified by the `BOTTOM_THICKNESS_VS_SI_WIDTH` statement, unless it contains the `CAPACITIVE_ONLY` keyword.

The following usage notes apply:

- The widths in these tables are post-etch widths, not drawn widths.
- The first width entry should be larger than or equal to the layer `WMIN` value.
- Every table must have at least two value pairs (width and corresponding RPSQ value).
- If you provide only one table of values, the `NUMBER_OF_TABLES` keyword and the table name are optional. However, if you use the keyword, you must also provide a table name, and vice versa.
- If you provide more than one table, the `NUMBER_OF_TABLES` keyword must be present and set to the actual number of tables. In addition, all tables must have a name.
- If you specify multiple `RPSQ_VS_SI_WIDTH` tables and multiple `GATE_TO_DIFFUSION_CAP` tables for the same conductor layer, the number of tables and the table names must be the same in both commands.
- If you specify multiple tables in a simultaneous multicorner extraction, the number of tables and the table names must be the same for each corner.
- When multiple `RPSQ_VS_SI_WIDTH` tables are specified on a ITF layer, you must specify which table to apply on a database layer in the layer mapping file with the following layer mapping syntax:

```
conducting_layers
     database_layer grd_layer [table_name=keyword]
```

The following table shows how the StarRC tool checks for table names in both the ITF and layer mapping files:

Table name in an ITF file	Table name in a layer mapping file	Behavior
N	N	The tool continues to lookup for the single table.

Feedback

Table name in an ITF file	Table name in a layer mapping file	Behavior
No	Yes	The tool ignores the table name in the layer mapping file, checks only for single table definition in the `RPSQ_VS_SI_WIDTH` table, and issues a warning message.
Yes	No	The tool is unable to extract resistance if the correct resistance setting is not found and issues an error message.
Yes	Yes	If the table name matches, the tool checks whether the same table name in the `RPSQ_VS_SI_WIDTH` table is specified in the layer mapping file. If the same table name is not specified, the tool issues an error message.

Errors

The following ITF commands are mutually exclusive methods for specifying conductor resistance: `RPSQ`, `RPSQ_VS_WIDTH_AND_SPACING`, `RPSQ_VS_SI_WIDTH`, `RHO`, `RHO_VS_SI_WIDTH_AND_THICKNESS`, and `RHO_VS_WIDTH_AND_SPACING`.

Examples

The following example specifies a single table of RPSQ values:

```
CONDUCTOR MET1 {
     THICKNESS=0.06 WMIN=0.02 SMIN=0.03
     RPSQ_VS_SI_WIDTH  {
          (0.01, 400) (0.02, 350)
          (0.03, 300) (0.04, 200)
     }
}
```

The following example specifies three tables of RPSQ values for the same conductor layer:

```
CONDUCTOR MET3 {
     THICKNESS=0.06 WMIN=0.01 SMIN=0.03
     RPSQ_VS_SI_WIDTH  {
          NUMBER_OF_TABLES = 3
          NGATE1 {
             (0.01, 400) (0.02, 350)
             (0.03, 300) (0.04, 200)
          }
          NGATE2 {
             (0.01, 325) (0.02, 300)
             (0.03, 275) (0.04, 250)
```

```
            }
            PGATE {
               (0.01, 600) (0.02, 550)
               (0.03, 500) (0.04, 340)
            }
}
```

The following example shows the `RPSQ_VS_SI_WIDTH` table with multiple table definitions:

```
CONDUCTOR GATE {
            THICKNESS=0.06650
            WMIN=0.01400
            SMIN=0.03400
            RPSQ_VS_SI_WIDTH {
               NUMBER_OF_TABLES = 4
               XYGATE   {
                          (0.014000,500.000000)
                          (0.017000,430.000000)
                        }
               BDGATE   {
                          (0.014000,564.000000)
                          (0.017000,530.000000)
                        }
               LMGATE_1 {
                          (0.014000,70.400000)
                          (0.017000,70.000000)
                        }
               XYGATE_2 {
                          (0.014000,90.000000)
                          (0.017000,85.000000)
                        }
            }
            LAYER_TYPE=GATE
}
```

The following example shows the information in the layer mapping file:

```
conducting_layers
            xygate GATE table_name=XYGATE
            bdgate GATE table_name=BDGATE
```

See Also

- ETCH_VS_WIDTH_AND_SPACING
- RPSQ
- RPSQ_VS_WIDTH_AND_SPACING

Feedback

RPSQ_VS_SI_WIDTH_AND_LENGTH

Models nonlinear gate resistances. Valid within a CONDUCTOR block.

Syntax

```
RPSQ_VS_SI_WIDTH_AND_LENGTH {
     WIDTHS  { w1  w2  … wn }
     LENGTHS { L1  L2  … Lm }
     VALUES  { v(w1,L1) v(w2,L1) ... v(wn,L1)
               v(w1,L2) v(w2,L2) ... v(wn,L2)
               ...
               v(w1,Lm) v(w2,Lm) ... v(wn,Lm)  }
     }
```

Arguments

Argument	Description
w1, w2, ...	Gate width values, in ascending order Units: microns
L1, L2,…	Gate length values, in ascending order Units: microns
v(w1,L1) ...	RPSQ values as a function of length and width Units: ohms per square The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Description

The RPSQ_VS_SI_WIDTH_AND_LENGTH command models nonlinear gate resistance.

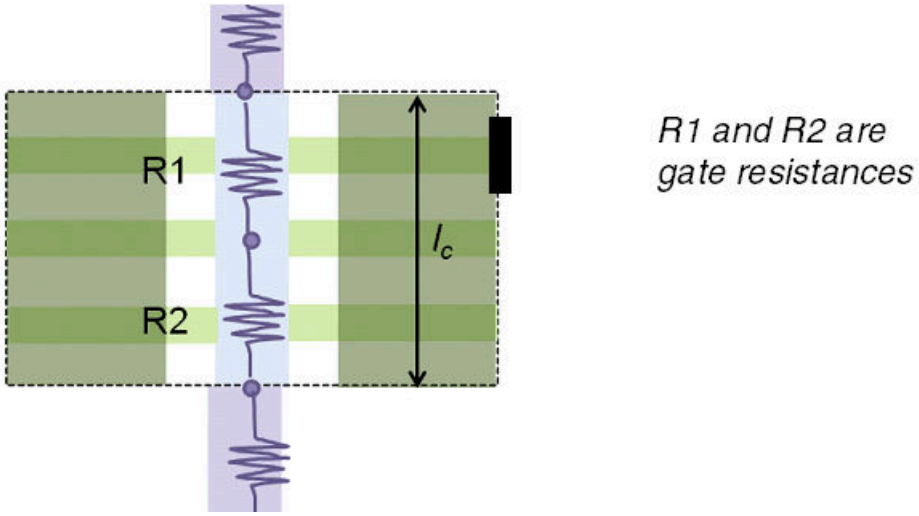
The conductor widths in this table are post-etch widths, also known as silicon width or physical width. The RPSQ values are the sheet resistance values that correspond to the silicon widths, not to drawn widths.

The RPSQ_VS_SI_WIDTH_AND_LENGTH command is valid only for a conductor layer whose LAYER_TYPE option is set to GATE. In addition, the ITF file must include another conductor for which the LAYER_TYPE option is set to FIELD_POLY.

The RPSQ value is also modified by the conductor thickness variation specified by the BOTTOM_THICKNESS_VS_SI_WIDTH statement, unless it contains the CAPACITIVE_ONLY keyword.

Feedback

Figure 269 FinFET Top View



The StarRC tool calculates the resistance as follows, where H is the gate layer height or thickness:

$R(W_n, L_m, H) = RPSQ_{n,m} * L_n / W_m$

If the conductor width is larger than the maximum gate width in the table, the StarRC tool uses the RPSQ value for the maximum gate width. If the conductor width is smaller than the minimum gate width in the table, the tool uses the RPSQ value for the minimum gate width.

Conductor length values outside the table range are handled in a similar manner. However, the actual conductor length and width values, not the boundary values, are used to calculate the resistance.

Examples

The following example demonstrates how to use the `RPSQ_VS_SI_WIDTH_AND_LENGTH` command to specify the RPSQ values shown in Table 102.

Table 102 Values for RPSQ Table

Gate width (microns)	RPSQ for length=0.05 um	RPSQ for length=0.1 um	RPSQ for length=0.15 um
0.02	0.1	0.23	0.45
0.022	0.12	0.26	0.47
0.024	0.18	0.28	0.53

Feedback

```
CONDUCTOR GATE {
THICKNESS = 0.6 WMIN = 0.3 SMIN = 0.15
LAYER_TYPE=GATE
RPSQ_VS_SI_WIDTH_AND_LENGTH {
   LENGTHS {0.05 0.1  0.15}
   WIDTHS {0.02 0.022 0.024}
   VALUES {0.1  0.23  0.45
           0.12 0.26  0.47
           0.18 0.28  0.53}
  }
}
```

See Also

- RPSQ_VS_SI_WIDTH
- RPSQ
- MULTIGATE

Feedback

RPSQ_VS_WIDTH_AND_SPACING

Specifies the resistance per square (RPSQ) for different conductor widths and spacings. Valid within a CONDUCTOR block.

Syntax

```
RPSQ_VS_WIDTH_AND_SPACING {
     SPACINGS {s1 s2 s3 ... sn }
     WIDTHS   {w1 w2 w3 ... wm }
     VALUES   {v(s1,w1) v(s2,w1) ... v(sn,w1)
              v(s1,w2) v(s2,w2) ... v(sn,w2)
                ...
              v(s1,wm) v(s2,wm) ... v(sn,wm)   }
}
```

Arguments

Argument	Description
s1 s2 ...	Spacings to the nearest conductor Units: microns
w1 w2 ...	Widths of the nearest conductor Units: microns
v(s1,w1) ...	Resistance per square values for the corresponding width and spacing Units: ohms per square

Description

The RPSQ_VS_WIDTH_AND_SPACING table models the effect of different conductor widths and spacings on the RPSQ value. Use this table to model process effects such as conductor cladding or dishing.

Note:

You can also specify the RPSQ value in the mapping file within the conducting_layers statement. The RPSQ value specified in the mapping file overrides any RPSQ, RPSQ_VS_WIDTH_AND_SPACING, or RPSQ_VS_SI_WIDTH statements in the ITF file.

The RPSQ value is also modified by the conductor thickness variation specified by the BOTTOM_THICKNESS_VS_SI_WIDTH statement, unless it contains the CAPACITIVE_ONLY keyword.

Feedback

The following usage notes apply:

- If the RPSQ value depends only on width, the SPACINGS index list can be skipped.
- If the RPSQ value depends only on spacing, the WIDTHS index list can be skipped.
- The first entry of the SPACINGS index list must be the same as the value of the SMIN keyword.
- The first entry of the WIDTHS index list must be the same as the value of the WMIN keyword.
- If the width or spacing falls between two indexes, the tool performs linear interpolation to calculate the RPSQ value.
- If the width or spacing falls outside the table boundaries, the tool uses the nearest boundary to calculate the RPSQ value.

Errors

The following options are mutually exclusive methods for specifying conductor resistance: RPSQ, RPSQ_VS_WIDTH_AND_SPACING, RPSQ_VS_SI_WIDTH, RHO, RHO_VS_SI_WIDTH_AND_THICKNESS, and RHO_VS_WIDTH_AND_SPACING.

Examples

```
CONDUCTOR m1 {
     THICKNESS = 0.6 WMIN = 0.25 SMIN = 0.25
     RPSQ_VS_WIDTH_AND_SPACING {
          SPACINGS {0.25 0.3}
          WIDTHS {0.25 0.3}
          VALUES {0.1 0.05 0.05 0.01}               }
     }
}
```

See Also

- ETCH_VS_WIDTH_AND_SPACING
- RPSQ
- RPSQ_VS_SI_WIDTH

Feedback

RPV

Specifies the resistive properties of a via layer. Valid within a `VIA` block.

Syntax

```
RPV = rpv_value
```

Arguments

Argument	Description
rpv_value	Resistance per via Units: ohms

Description

The resistive properties of the via layer must be specified. The via resistance can be specified in one of three mutually exclusive ways: `RHO`, `RPV` and `AREA`, or `RPV_VS_AREA`.

If `RPV` is specified, `AREA` is also required.

The default of `RPV` is such that RPV x AREA = 1-0e-6.

Examples

```
VIA via1 {
     FROM=m1 TO=m2 AREA=0.5 RPV=4
}
```

See Also

- RHO
- RPV_VS_AREA

Feedback

RPV_ADJUSTMENT_FACTOR_VS_VIA_COUNT

Specifies an adjustment to the resistance of a merged via array with respect to the via count. Valid within a `VIA` block for transistor-level flows.

Syntax

```
RPV_ADJUSTMENT_FACTOR_VS_VIA_COUNT {
     VIA_SIZE   {ax ay}
     NUMBER_OF_VIAS { an1, an2, … anm }
     VALUE {av1, av2, … avn}
     [VIA_SIZE   {bx by}
     NUMBER_OF_VIAS { bn1, bn2, … bnm }
     VALUE {bv1, bv2, … bvn}
     …
    }
```

Arguments

Argument	Description
ax ay, *bx by*, ...	Via size in the x- and y-dimensions (as-drawn dimensions) Units: microns
an1, an2, …bn1, bn2, …	Number of vias in the original (unmerged) array Units: none
av1, av2, …bv1, bv2, …	Adjustment factor (multiplier) with respect to the corresponding via count and via size Units: none

Description

The `RPV_ADJUSTMENT_FACTOR_VS_VIA_COUNT` command specifies an adjustment factor for the resistance of a merged via array.

In transistor-level flows, via merging is controlled by the `MERGE_VIAS_IN_ARRAY` command in the StarRC command file and the `MAX_VIA_ARRAY_LENGTH` and `MAX_VIA_ARRAY_SPACING` parameters in the `via_layers` section of the mapping file, as shown in Table 103.

Table 103 Via Merging Behavior For Transistor-Level Flows

MERGE_VIAS_IN_ARRAY	Parameters are set in mapping file	Parameters are not set in mapping file
YES	Merge via array based on mapping file parameters	Merge via array to an extent calculated by internal heuristics

Feedback

Table 103 Via Merging Behavior For Transistor-Level Flows (Continued)

MERGE_VIAS_IN_ARRAY	Parameters are set in mapping file	Parameters are not set in mapping file
NO (or not set)	Merge via array based on mapping file parameters	Do not merge vias

After merging, the StarRC tool extracts one resistance for the merged via array, based on calculating the parallel resistance of the original vias.

The `RPV_ADJUSTMENT_FACTOR_VS_VIA_COUNT` option allows you to specify a multiplier to apply to the extracted resistance. For a single via size, you specify a list of via counts (from the original unmerged array) and a corresponding list of factors. You can specify factors for multiple via sizes by repeating the keywords.

The following usage notes apply:

- The via dimensions in the `VIA_SIZE` keyword must exactly match the layout via dimensions, either as drawn or after rotating 90-degrees. If a via does not match any of the specified via sizes, the tool does not apply an adjustment factor to the via resistance.
- If the number of vias is between two values in the `NUMBER_OF_VIAS` list, the tool performs linear interpolation.
- If the number of vias is outside the range of the `NUMBER_OF_VIAS` list, the tool uses the nearest boundary value.
- Adjustment tables must be consistent between corners in SMC runs. If a table exists for one corner, it must exist for all corners. The contents of the `VIA_SIZE` and `NUMBER_OF_VIAS` keywords must be the same for all corners. However, the adjustment factors can be different for different corners.

Examples

The following command provides tables for two via sizes:

```
RPV_ADJUSTMENT_FACTOR_VS_VIA_COUNT {
     VIA_SIZE { 0.032 0.032 }
        NUMBER_OF_VIAS { 2 8 }
        VALUE { 1.1 1.24 }
     VIA_SIZE { 0.064 0.64 }
        NUMBER_OF_VIAS { 2 8 }
        VALUE { 1.05 1.15 }
```

Feedback

See Also

- MERGE_VIAS_IN_ARRAY
- via_layers

Feedback

RPV_VS_AREA

Specifies the resistance per via (RPV) as a function of via area. Valid within a VIA block.

Syntax

```
RPV_VS_AREA {
     (area1, rpv1)
     (area2, rpv2)
     (area3, rpv3)
     …
}
```

Arguments

Argument	Description
area	Via area Units: square microns
rpv	Resistance per via Units: ohms

Description

Use the RPV_VS_AREA table in a VIA statement to specify the via resistance as a function of via size. The RPV_VS_AREA cannot be used with RPV or RHO in the same VIA block.

Use one of the following mutually exclusive methods to define via resistance:

- Specify the bulk resistivity with the RHO statement. The resistance of a via is calculated as

  $$R = \frac{\rho \times t}{l \times w}$$

  where ρ is the bulk resistivity, t is the via thickness (height), and l and w are the lateral via length and width.

- Specify the resistance per via with the RPV statement and the via area with the AREA statement. The resistance of a via having an arbitrary area is calculated as

  $$R = \frac{RPV \times AREA}{l \times w}$$

  where l and w are the lateral via length and width.

- Specify a table of values with the RPV_VS_AREA statement.

RPV and AREA values specified in a mapping file take precedence over the `RPV_VS_AREA` values specified in an ITF file.

Interpolation and Extrapolation

For most vias, interpolation and extrapolation are handled as follows:

- If the via area falls between two area entries in the `RPV_VS_AREA` table, the StarRC tool converts resistance to conductance, performs linear interpolation in the conductance domain, then converts conductance to resistance.
- If the via area is less than the smallest area entry in the `RPV_VS_AREA` table, the tool uses the RPV value for the smallest area entry in the table.
- If the via area is greater than the largest area entry in the `RPV_VS_AREA` table, the tool uses the RPV value for the largest area entry in the table.

Trench contacts can have tall covertical layers that are not connected by physical vias. To extract vertical resistance, virtual vias are inserted between the trench contact conductors. The StarRC tool segments these vias automatically to create a distributed resistance network. For trench contact virtual vias, interpolation and extrapolation are handled as follows:

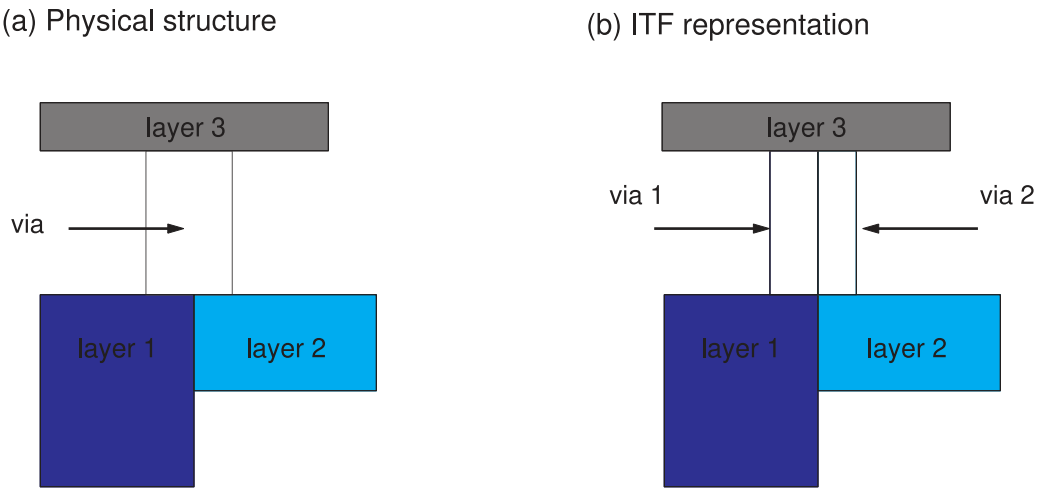
- If the via area falls between two area entries in the `RPV_VS_AREA` table, the StarRC tool converts resistance to conductance, performs linear interpolation in the conductance domain, then converts conductance to resistance.
- If the via area is less than the smallest area entry in the `RPV_VS_AREA` table, the tool determines the conductance of the via with the smallest area entry and uses that value to calculate the via resistance for any vias with smaller areas.
- If the via area is greater than the largest area entry in the `RPV_VS_AREA` table, the tool determines the conductance of the via with the largest area entry and uses that value to calculate the via resistance for any vias with larger areas.

Abutting Vias

Physically, a single via might land on the boundary between two conductor layers. However, ITF process description rules require that a via be defined between one upper layer and one lower layer. Therefore the nxtgrd file represents the physical via as two abutting vias, with one via for each conductor layer, as shown in Figure 270.

Feedback

Figure 270 Abutting Vias



The StarRC tool handles abutting vias in the presence of `RPV_VS_AREA` tables as follows:

- Adds the areas of the two vias
- Uses the merged area to calculate the total resistance using the `RPV_VS_AREA` table
- Distributes the total resistance between the original vias based on the original via areas

Guidelines:

- You should ensure that the `RPV_VS_AREA` tables in the separate `VIA` blocks are identical. The StarRC tool does not check for equivalency.
- This procedure does not apply to trench contact fake vias or to vias that are overlapping instead of abutting.
- This procedure applies only to transistor-level extraction.

Examples

```
VIA via1 {
    FROM=m1 TO=m2
    RPV_VS_AREA { (200, 0.5) (350, 0.5) (600, 0.25) }
}
```

Feedback

RPV_VS_COVERAGE

Specifies resistance per via (RPV) values with respect to via size, upper layer coverage, and lower layer coverage. Valid within a `VIA` block.

Syntax

```
RPV_VS_COVERAGE [UPPER_LAYER_ONLY | LOWER_LAYER_ONLY] {
     VIA_SIZE   {sx1 sy1}
     COV_X { cx1, cx2, … cxm }
     COV_Y {cy1, cy2, … cyn}
     VALUES {r(cx1,cy1) r(cx1,cy2) … r(cx1,cyn)
             r(cx2,cy1) r(cx2,cy2) … r(cx2,cyn)
             …
             r(cxm,cy1) r(cxm,cy2) … r(cxm,cyn)  }
     [VIA_SIZE   {sx2 sy2}
     COV_X { dx1, dx2, … dxm }
     COV_Y {dy1, dy2, … dyn}
     VALUES {r(dx1,dy1) r(dx1,dy2) … r(dx1,dyn)
             r(dx2,dy1) r(dx2,dy2) … r(dx2,dyn)
             …
             r(dxm,dy1) r(dxm,dy2) … r(dxm,dyn)  } ]
     …
    }
```

Arguments

Argument	Description
`UPPER_LAYER_ONLY`	Optional; restricts the coverage check to the upper layer
`LOWER_LAYER_ONLY`	Optional; restricts the coverage check to the lower layer
sx1 sy1	Via size in the x- and y-dimensions (as-drawn dimensions) Units: microns
cx1, cx2, …dx1, dx2, …	X-direction coverage values, in ascending order; coverage values for different via sizes do not have to match Units: microns
cy1, cy2, …dy1, dy2, …	Y-direction coverage values, in ascending order; coverage values for different via sizes do not have to match Units: microns
r(cx1,cy1) r(dx1,dy1)	Resistance per via (RPV) for the corresponding coverage values Units: ohms

Feedback

Description

The RPV_VS_COVERAGE command specifies resistance per via (RPV) values based on the via size and the amount of upper and lower conductor layer coverage.

Note:

The RPV_VS_COVERAGE command uses as-drawn dimensions for the via and for the upper and lower layer coverage features. You can use post-etch dimensions for the upper layer coverage analysis by using the RPV_VS_SI_COVERAGE command instead.

The RPV_VS_COVERAGE command contains one or more tables of RPV values. Each table applies to a specific via size, denoted by the VIA_SIZE keyword, based on drawn dimensions. The via dimensions in the VIA_SIZE keyword must exactly match the layout via dimensions, either as drawn or after rotating 90-degrees.

If a via does not match any of the via sizes specified in the RPV_VS_COVERAGE command, the StarRC tool uses the RPV_VS_AREA command to determine the via resistance.

The StarRC tool first checks to see if the layout via dimensions (x,y) match any entries in the VIA_SIZE list. If a match exists, the layout coverage dimensions are used to determine the COV_X and COV_Y indexes and subsequently the RPV value from the VALUES table. If there is no match with the original layout, the tool uses the rotated dimensions (y,x) to see if there is a match to any entries in the VIA_SIZE list. If a match is found, the new (rotated) coverage dimensions are used to determine the RPV value.

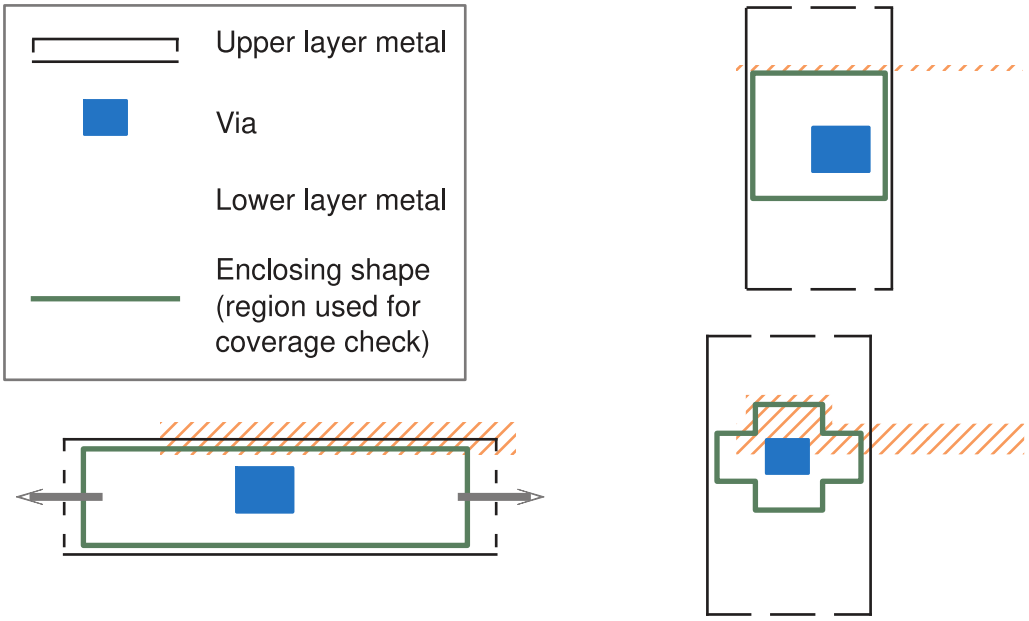
When a via matches a via size specified by a VIA_SIZE keyword, the StarRC tool determines the via resistance using the following procedure:

1. Find the enclosing shape that covers the via.

   The tool performs a Boolean AND operation of the upper conductor layer and lower conductor layer to get the enclosing shape that covers the via, as shown in Figure 271.

Feedback

Figure 271 Via Coverage Enclosing Shape



You can optionally use only the upper layer for the coverage check by specifying the `UPPER_LAYER_ONLY` keyword immediately after the `RPV_VS_COVERAGE` command. Similarly, the `LOWER_LAYER_ONLY` keyword specifies to use only the lower layer.

2. Decompose the enclosing shape into rectangular coverage boxes and compare them.

   A complex enclosing shape is treated as a set of overlapping boxes, as shown in Figure 272. The `COV_X` and `COV_Y` values in the `RPV_VS_COVERAGE` command define a set of coverage boxes with those X and Y dimensions. (The number of available boxes is the product of the number of `COV_X` values and the number of `COV_Y` values.)

   The StarRC tool determines the largest coverage box that, when centered over the via, fits within the enclosing shape.

   If multiple coverage boxes fit within the enclosing shape, the box with the largest minimum dimension is selected, as shown in Figure 272. If multiple coverage boxes have the same minimum dimension values, the box with the largest maximum dimension is selected. An example is shown in Figure 273.

Feedback

Figure 272 Coverage Rule Selection Example 1

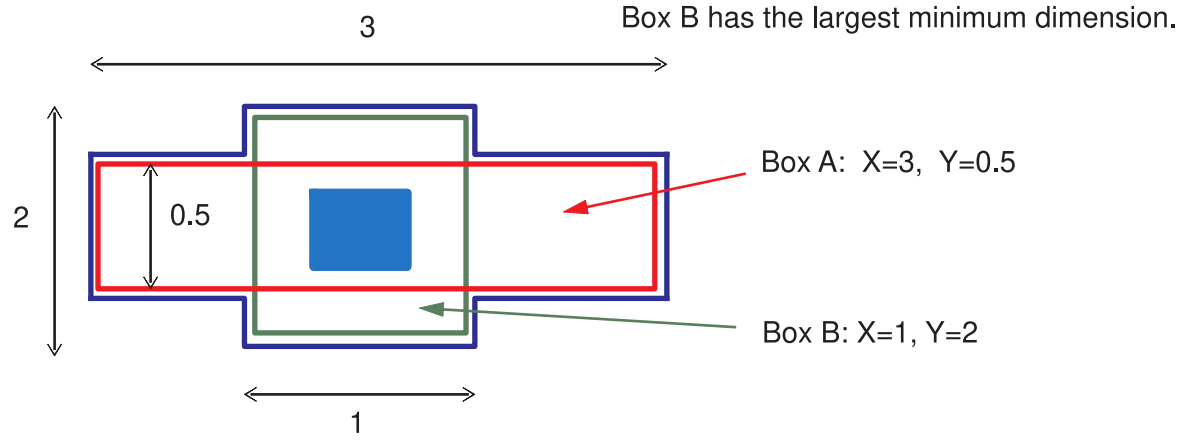
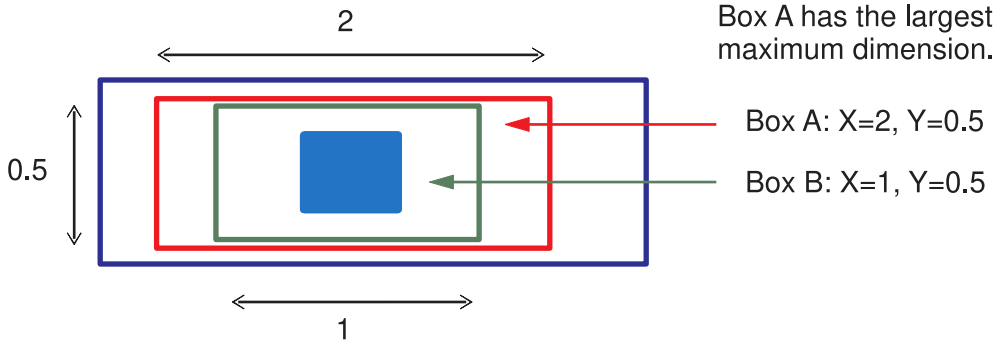


Figure 273 Coverage Rule Selection Example 2



3. Look up or calculate the resistance per via.

   The possible configurations are as follows, assuming that the coverage box is centered over the via.

   - If all of the coverage boxes are too large to fit within the enclosing shape, the tool uses the RPV value of the smallest coverage box.
   - If all of the coverage boxes fit within the enclosing shape, but the enclosing shape extends beyond the largest coverage box in both X and Y dimensions, the tool uses the RPV value of the largest coverage box.
   - If an edge of the enclosing shape falls between the edges of two coverage boxes, linear interpolation is used to calculate the RPV value. Interpolation in both the X and Y dimensions might be necessary.

Feedback

A single via definition in the ITF file can have either one or two RPV_VS_COVERAGE tables. Valid combinations are as follows:

- One table, can be any one of the following:
  - RPV_VS_COVERAGE
  - RPV_VS_COVERAGE UPPER_LAYER_ONLY
  - RPV_VS_COVERAGE LOWER_LAYER_ONLY
- Two tables, must be both of the following:
  - RPV_VS_COVERAGE UPPER_LAYER_ONLY
  - RPV_VS_COVERAGE LOWER_LAYER_ONLY

If two RPV_VS_COVERAGE tables are present, one must have the UPPER_LAYER_ONLY keyword while the other must have the LOWER_LAYER_ONLY keyword. In this case, the StarRC tool finds the RPV values from both tables and uses the lowest RPV value.

Similarly, a single via definition can have either one or two RPV_VS_SI_COVERAGE tables, with the same valid combinations of upper and lower layer options.

A single via cannot have both an RPV_VS_SI_COVERAGE table and an RPV_VS_COVERAGE table.

Examples

Example of Coverage Analysis

The following command provides one table for vias of size 2 x 2.The structure in Figure 274 illustrates how this table might be applied.

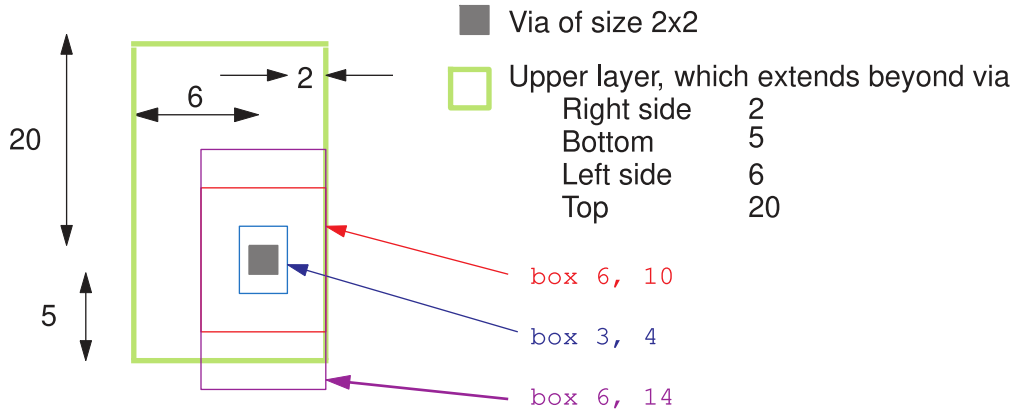
```
RPV_VS_COVERAGE {
     VIA_SIZE { 2 2 }
        COV_X { 3 6 8 }
        COV_Y { 4 10 14 }
        VALUES { 18 20 25 21 26 30 27 32 35 }
}
```

The table defines nine coverage boxes based on the COV_X and COV_Y values. The RPV values in the VALUES list correspond to the X,Y pairs in this order: (3,4) (3,10) (3,14) (6,4) (6,10) (6,14) (8,4) (8,10) (8,14).

In Figure 274, the via is positioned near the lower-right corner of the upper level feature. The coverage boxes with X dimension 6 exactly match the nearest X edge of the upper layer feature, therefore boxes with other X dimensions are eliminated from consideration.

Feedback

Figure 274 Coverage Box Usage



The nearest Y edge of the upper layer feature falls between the edges of the (6,10) and (6,14) coverage boxes. Linear interpolation is used between the RPV values of these two coverage boxes to calculate the RPV value for this case. If the (6,14) coverage box were not included in the table, the RPV value of the (6,10) coverage box would be used. In other words, no extrapolation is performed.

Example With Multiple Tables

The following command provides tables for two via sizes:

```
RPV_VS_COVERAGE {
     VIA_SIZE { 0.032 0.032 }
        COV_X { 0.064, 0.096, 0.2 }
        COV_Y { 0.064, 0.096, 0.2 }
        VALUES { 18 20 25 21 26 30 27 32 35 }
     VIA_SIZE { 0.064 0.64 }
        COV_X { 0.064, 0.096, 0.2 }
        COV_Y { 0.064, 0.096, 0.2 }
        VALUES { 6 8 13 9 14 18 15 20 23 }    }
```

See Also

- RPV_VS_AREA
- RPV_VS_SI_COVERAGE

Feedback

RPV_VS_SI_COVERAGE

Specifies resistance per via (RPV) values with respect to via size and upper layer coverage, taking etch effects into account. Valid within a `VIA` block.

Syntax

```
RPV_VS_SI_COVERAGE [UPPER_LAYER_ONLY | LOWER_LAYER_ONLY] {
     VIA_SIZE   {sx1 sy1}
     COV_X { cx1, cx2, … cxm }
     COV_Y {cy1, cy2, … cyn}
     VALUES {r(cx1,cy1) r(cx1,cy2) … r(cx1,cyn)
             r(cx2,cy1) r(cx2,cy2) … r(cx2,cyn)
             …
             r(cxm,cy1) r(cxm,cy2) … r(cxm,cyn)    }
     [VIA_SIZE   {sx2 sy2}
     COV_X { dx1, dx2, … dxm }
     COV_Y {dy1, dy2, … dyn}
     VALUES {r(dx1,dy1) r(dx1,dy2) … r(dx1,dyn)
             r(dx2,dy1) r(dx2,dy2) … r(dx2,dyn)
             …
             r(dxm,dy1) r(dxm,dy2) … r(dxm,dyn)    }
     ]
     …
    }
```

Arguments

Argument	Description
`UPPER_LAYER_ONLY`	Optional; restricts the coverage check to the upper layer
`LOWER_LAYER_ONLY`	Optional; restricts the coverage check to the lower layer
sx1 sy1	Via size in the x- and y-dimensions (as-drawn dimensions) Units: microns
cx1, cx2, …dx1, dx2, …	X-direction coverage values, in ascending order; coverage values for different via sizes do not have to match Units: microns
cy1, cy2, …dy1, dy2, …	Y-direction coverage values, in ascending order; coverage values for different via sizes do not have to match Units: microns
r(cx1,cy1) r(dx1,dy1)	Resistance per via (RPV) for the corresponding coverage values Units: ohms

Feedback

Description

The RPV_VS_SI_COVERAGE command specifies resistance per via (RPV) values based on the via size and the amount of upper conductor layer coverage.

Note:

The RPV_VS_SI_COVERAGE command uses post-etch dimensions for the upper layer coverage features. You can use as-drawn dimensions by using the RPV_VS_COVERAGE command instead.

The RPV_VS_SI_COVERAGE command contains one or more tables of RPV values. Each table applies to a specific via size, denoted by the VIA_SIZE keyword, based on drawn dimensions. The via dimensions in the VIA_SIZE keyword must exactly match the layout via dimensions, either as drawn or after rotating 90-degrees.

If a via does not match any of the via sizes specified in the RPV_VS_SI_COVERAGE command, the StarRC tool uses the RPV_VS_AREA command to determine the via resistance.

The StarRC tool first checks to see if the layout via dimensions (x,y) match any entries in the VIA_SIZE list. If a match exists, the layout coverage dimensions are used to determine the COV_X and COV_Y indexes and subsequently the RPV value from the VALUES table. If there is no match with the original layout, the tool uses the rotated dimensions (y,x) to see if there is a match to any entries in the VIA_SIZE list. If a match is found, the new (rotated) coverage dimensions are used to determine the RPV value.

When a via matches a via size specified by a VIA_SIZE keyword, the tool calculates the via resistance according to the following procedure.

1. Apply etch values to the upper or lower layer feature that covers the via.

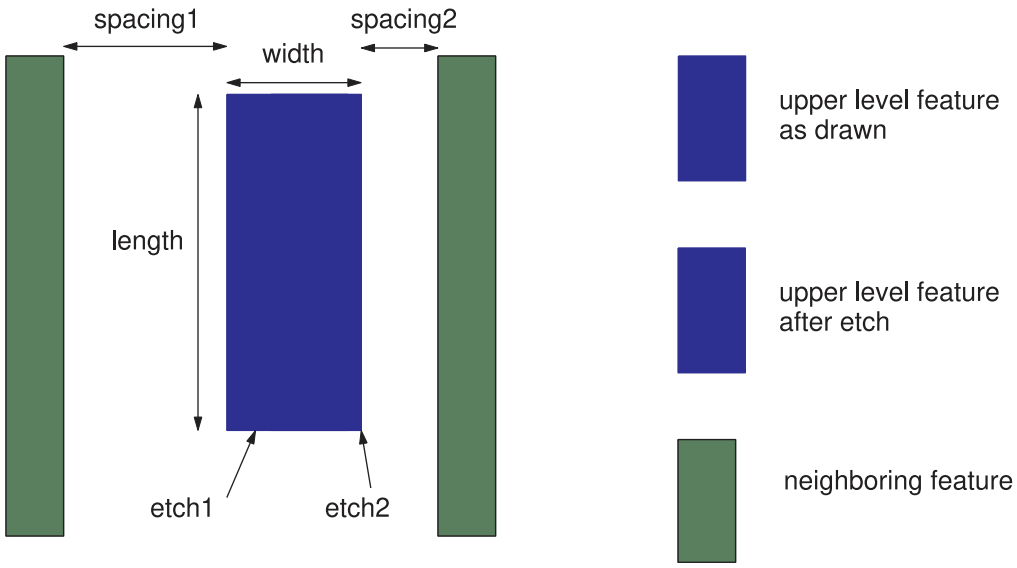
   You can optionally use only the upper layer for the coverage check by specifying the UPPER_LAYER_ONLY keyword immediately after the RPV_VS_COVERAGE command. Similarly, the LOWER_LAYER_ONLY keyword specifies to use only the lower layer.

   Figure 275 illustrates the etch effect. The upper level feature covering the via is shown in light blue for the as-drawn dimensions and dark blue for the post-etch dimensions. A neighboring feature is present on each side, with different spacings on the two sides. If the ETCH_VS_WIDTH_AND_SPACING command is used to specify the etch conditions, the amount of etch applied to the upper level feature might be different on each side based on the spacing to the closest neighboring feature.

   As a simplification, the StarRC tool calculates the average of the actual etch values on the two sides of the upper layer feature and applies the average etch to both sides.

Feedback

Figure 275 Example of Feature Etch



2. Evaluate the available coverage boxes.

   The `COV_X` and `COV_Y` values in the `RPV_VS_SI_COVERAGE` command define a set of coverage boxes with those X and Y dimensions. (The number of available boxes is the product of the number of `COV_X` values and the number of `COV_Y` values.)

   The StarRC tool determines the largest coverage box that, when centered over the via, fits within the etched upper layer feature.

3. Look up or calculate the resistance per via.

   The possible configurations are as follows, assuming that the coverage box is centered over the via.

   - If all of the coverage boxes are too large to fit within the etched upper layer feature, the tool uses the RPV value of the smallest coverage box.
   - If all of the coverage boxes fit within the upper layer feature, but the upper layer extends beyond the largest coverage box in both X and Y dimensions, the tool uses the RPV value of the largest coverage box.
   - If an edge of the upper layer feature falls between the edges of two coverage boxes, linear interpolation is used to calculate the RPV value. Interpolation in both the X and Y dimensions might be necessary.

Feedback

A single via definition in the ITF file can have either one or two `RPV_VS_SI_COVERAGE` tables. Valid combinations are as follows:

- One table, can be any one of the following:
  - `RPV_VS_SI_COVERAGE`
  - `RPV_VS_SI_COVERAGE UPPER_LAYER_ONLY`
  - `RPV_VS_SI_COVERAGE LOWER_LAYER_ONLY`
- Two tables, must be both of the following:
  - `RPV_VS_SI_COVERAGE UPPER_LAYER_ONLY`
  - `RPV_VS_SI_COVERAGE LOWER_LAYER_ONLY`

If two `RPV_VS_SI_COVERAGE` tables are present, one must have the `UPPER_LAYER_ONLY` keyword while the other must have the `LOWER_LAYER_ONLY` keyword. In this case, the StarRC tool finds the RPV values from both tables and uses the lowest RPV value.

Similarly, a single via definition can have either one or two `RPV_VS_COVERAGE` tables, with the same valid combinations of upper and lower layer options.

A single via cannot have both an `RPV_VS_SI_COVERAGE` table and an `RPV_VS_COVERAGE` table.

See Also

- RPV_VS_AREA
- RPV_VS_COVERAGE

Feedback

RPV_VS_SI_WIDTH_AND_LENGTH

Models the resistivity of a trench contact virtual via according to the silicon width and length of one of the associated trench contact conductor layers. Valid within a VIA block.

Syntax

```
ETCH_ASSOCIATED_LAYER = tc_layer
RPV_VS_SI_WIDTH_AND_LENGTH {
   LENGTHS { L1 L2  ... Ln }
   WIDTHS  { w1 w2  ... wm }
   VALUES  { r(L1,w1) r(L2,w1)  ... r(Ln,w1)
             r(L1,w2) r(L2,w2)  ... r(Ln,w2)
             ...
             r(L1,wm) r(L2,wm)  ... r(Ln,wm)
           }
}
```

Arguments

Argument	Description
tc_layer	Either the FROM or TO layer in the VIA definition; must be a trench contact layer defined with the LAYER_TYPE=TRENCH_CONTACT statement
L1, L2, ...	Via lengths, in ascending order Units: microns
w1, w2, ...	Via widths, in ascending order Units: microns
r(L1,w1), r(L2,w1), ...	Resistance per via Units: ohms

Description

In trench contact layers, electrical current flows both along the routing direction and also in the vertical direction. As a result, the contact resistance behavior can be complex.

Feedback

You can model the trench contact resistance as follows:

- Use the VIA statement to define a virtual via between two trench contact layers or between a trench contact layer and another conductor layer.
- Specify the variation of via resistance with respect to the drawn length and width dimensions by using the RPV_VS_WIDTH_AND_LENGTH statement in the via definition.
- Specify the variation of via resistance with respect to the silicon length and width dimensions (in other words, the dimensions after applying etch operations) by using the RPV_VS_SI_WIDTH_AND_LENGTH statement in the via definition.

The resistance is calculated as follows:

- If the length and width of the trench contact via fall within the bounds of the values in the LENGTHS and WIDTHS argument lists, linear interpolation is applied between the nearest values.
- If the length is smaller than the minimum value in the LENGTHS argument list, the StarRC tool finds the resistance value that corresponds to the minimum value in the LENGTHS argument list and scales the resistance in proportion to the minimum length divided by the actual length. Width values smaller than the minimum value in the WIDTHS argument list are handled in the same way. In other words,

$$R_{new} = R_{min} \cdot \frac{W_{min} \cdot L_{min}}{W_{actual} \cdot L_{actual}}$$

- If the length is larger than the maximum value in the LENGTHS argument list, the StarRC tool finds the resistance value that corresponds to the maximum value in the LENGTHS argument list and scales the resistance in proportion to the maximum length divided by the actual length. Width values larger than the maximum value in the WIDTHS argument list are handled in the same way. In other words,

$$R_{new} = R_{max} \cdot \frac{W_{max} \cdot L_{max}}{W_{actual} \cdot L_{actual}}$$

RPV values represent the resistance of the entire virtual via regardless of any segmentation specified by the TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO command in the StarRC command file.

Feedback

Figure 276 shows a trench contact layer and the virtual via that connects it to the diffusion layer. The length dimension of the trench contact is parallel to the adjacent transistor gate polygon.

Figure 276 As-Drawn Trench Contact Virtual Via

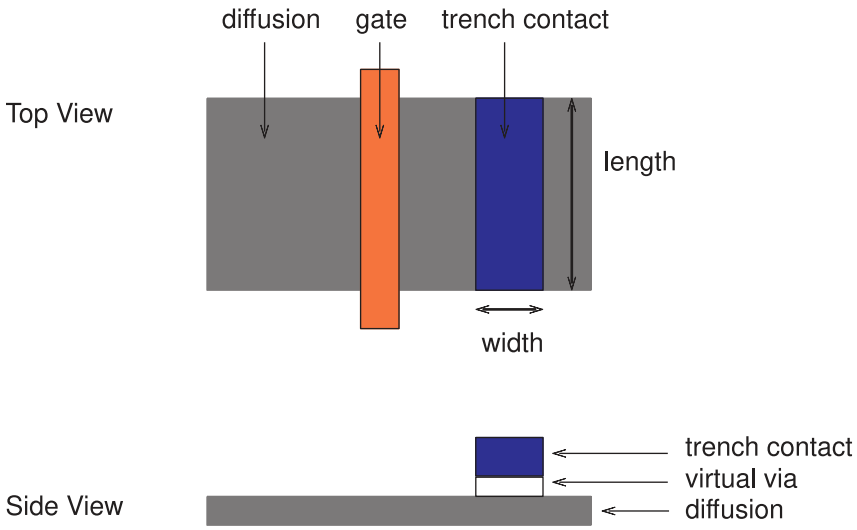
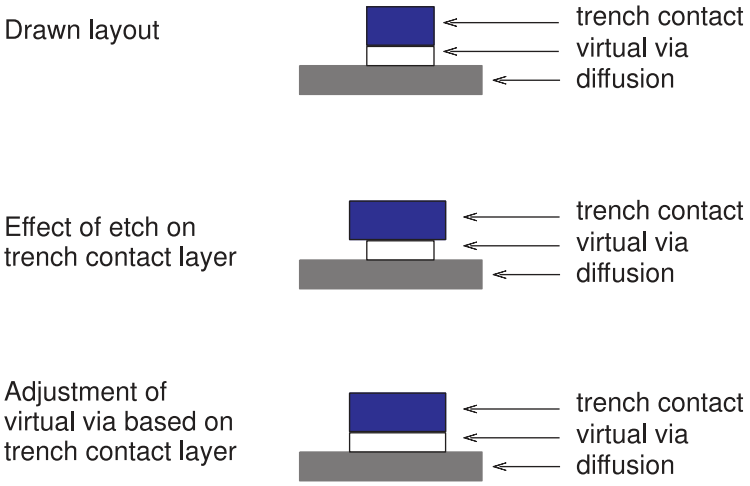


Figure 277 shows how a trench contact conductor layer might change after an etch operation. For accurate modeling, the trench contact virtual via dimensions must match the post-etch (silicon) dimensions of the `FROM` or `TO` layers associated with the via. The `ETCH_ASSOCIATED_LAYER` keyword specifies which of the two layers the virtual via dimensions should match.

Feedback

Figure 277 Post-Silicon Trench Contact Virtual Via



To use the drawn dimensions of the virtual via instead of the silicon dimensions, use the `RPV_VS_WIDTH_AND_LENGTH` statement in the via definition instead of the `RPV_VS_SI_WIDTH_AND_LENGTH` statement.

Examples

In the following example, the trench contact virtual via VTC is defined between layers POD and M0. The via dimensions are set to match the dimensions of layer M0 after the application of the etch operation defined by the `ETCH_VS_WIDTH_AND_SPACING` command.

```
CONDUCTOR M0 {THICKNESS=0.9 WMIN=0.09 SMIN=0.09 LAYER_TYPE=TRENCH_CONTACT
     ETCH_VS_WIDTH_AND_SPACING { ... }
}
VIA VTC { FROM=POD TO=M0
     ETCH_ASSOCIATED_LAYER = M0
     RPV_VS_SI_WIDTH_AND_LENGTH {
        LENGTHS { 0.03 0.04 0.08 0.12 0.6 }
        WIDTHS { 0.02 0.03 0.04 }
        VALUES {
           360 160 80 60 22
           260 90 80 50 21
           200 80 70 40 20
        }
     }
}
```

See Also

- VIA
- RPV_VS_WIDTH_AND_LENGTH

- ETCH_VS_WIDTH_AND_SPACING
- TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO

Feedback

RPV_VS_WIDTH_AND_LENGTH

Models the resistivity of a trench contact virtual via according to the drawn width and length of one of the associated trench contact conductor layers. Valid within a VIA block.

Syntax

```
RPV_VS_WIDTH_AND_LENGTH {
   LENGTHS { L1 L2  ... Ln }
   WIDTHS  { w1 w2  ... wm }
   VALUES  { r(L1,w1) r(L2,w1)  ... r(Ln,w1)
             r(L1,w2) r(L2,w2)  ... r(Ln,w2)
             ...
             r(L1,wm) r(L2,wm)  ... r(Ln,wm)
           }
}
```

Arguments

Argument	Description
L1, L2, …	Via lengths, in ascending order Units: microns
w1, w2, …	Via widths, in ascending order Units: microns
r(L1,w1), r(L2,w1), …	Resistance per via or RPV Units: ohms

Description

In trench contact layers, electrical current flows both along the routing direction and in the vertical direction. You can model the vertical current flow as follows:

- Use the VIA statement to define a virtual via between two trench contact layers or between a trench contact layer and another conductor layer.
- Specify the variation of via resistance with respect to the drawn length and width dimensions by using the RPV_VS_WIDTH_AND_LENGTH statement in the via definition.
- Specify the variation of via resistance with respect to the silicon length and width dimensions (in other words, the dimensions after applying etch operations) by using the RPV_VS_SI_WIDTH_AND_LENGTH statement in the via definition.

The resistance is calculated as follows:

- If the length and width of the trench contact via fall within the bounds of the values in the LENGTHS and WIDTHS argument lists, linear interpolation is applied between the nearest values.
- If the length is smaller than the minimum value in the LENGTHS argument list and the width is smaller than the minimum value in the WIDTHS argument list, the StarRC tool finds the resistance that corresponds to the minimum length and width values and scales the resistance in proportion to the minimum dimensions divided by the actual dimensions. In other words,

$$R_{new} = R_{min} \cdot \frac{W_{min} \cdot L_{min}}{W_{actual} \cdot L_{actual}}$$

- If the length is larger than the maximum value in the LENGTHS argument list and the width is larger than the maximum value in the WIDTHS argument list, the StarRC tool finds the resistance that corresponds to the maximum length and width values and scales the resistance in proportion to the maximum dimensions divided by the actual dimensions. In other words,

$$R_{new} = R_{max} \cdot \frac{W_{max} \cdot L_{max}}{W_{actual} \cdot L_{actual}}$$

- If only one dimension (width or length) is out of bounds, the StarRC tool performs similar scaling for the out-of-bounds dimension while using linear interpolation for the other dimension. For example, if the width is smaller than the minimum specified width, but the length falls within the specified range, the resistance is calculated as

$$R_{new} = R(Wmin, L) \cdot \frac{W_{min}}{W_{actual}}$$

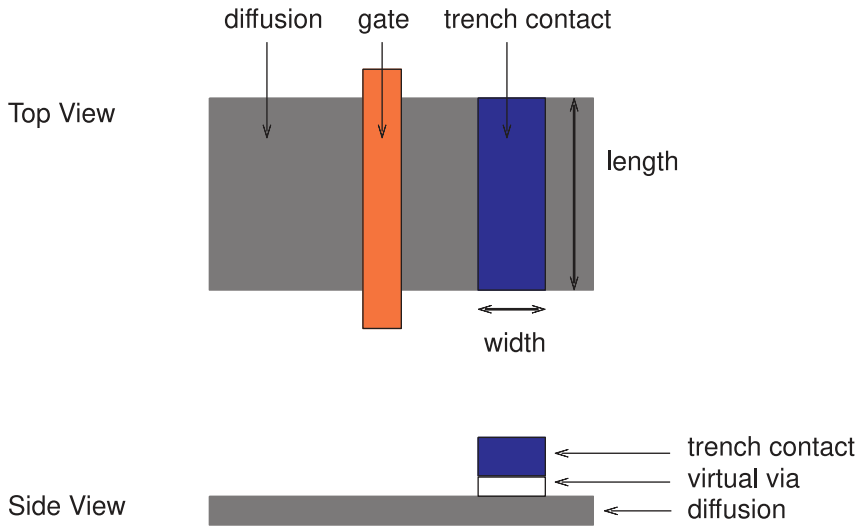
RPV values represent the resistance of the entire virtual via regardless of any segmentation specified by the TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO command in the StarRC command file.

The resistance of the virtual via is dependent on its length and width. If the top conductor layer is affected by etch operations, the virtual via size should also change to match the final silicon dimensions of the conductor layer. To use the silicon dimensions instead of the drawn dimensions, use the RPV_VS_SI_WIDTH_AND_LENGTH statement instead of the RPV_VS_WIDTH_AND_LENGTH statement.

Feedback

Figure 278 shows a trench contact layer and the virtual via that connects it to the diffusion layer. The length dimension of the trench contact is parallel to the adjacent transistor gate polygon.

Figure 278 As-Drawn Trench Contact Virtual Via



Examples

In the following example, the trench contact virtual via VTC is defined between layers POD and M0. The via dimensions are the as-drawn dimensions and are not changed by any etch operations on the associated conductors.

```
VIA VTC { FROM=POD TO=M0 }
RPV_VS_WIDTH_AND_LENGTH {
   LENGTHS { 0.02 0.04 0.05 }
   WIDTHS {0.01]
   VALUES { 100 80 60 }
}
```

See Also

- VIA
- RPV_VS_SI_WIDTH_AND_LENGTH
- TRENCH_CONTACT_VIRTUAL_VIA_SEGMENTATION_RATIO

Feedback

RVTV

Identifies specific conductors that connect a backside metal layer to a frontside metal layer.

Syntax

```
RVTV rvtv_name {
  [DIFFUSION_TO_RVTV_SMIN = spacing]
  [GATE_TO_CONTACT_SMIN = spacing]
  SMIN = min_spacing
  WMIN = min_width
  ...
}
```

Arguments

Argument	Description
rvtv_name	Name of the RVTV conductor or dielectic layer
DIFFUSION_TO_RVTV_SMIN	Minimum distance between the diffusion and RVTV conductor. (valid for BM0 to M0 or trench contact connection) Units: microns Default: SMIN
GATE_TO_CONTACT_SMIN	Minimum distance between the gate and RVTV conductor (valid for BM0 to diffusion connection) Units: microns Default: SMIN
min_spacing	Minimum spacing between two geometries on this layer Units: microns
min_width	Minimum width of a geometry on this layer Units: microns

Description

The RVTV statement allows you to identify conductors that connect a backside metal layer to a frontside metal layer. You can use the same syntax or parameters of the CONDUCTOR statement when you use the RVTV statement.

For detailed information about the CONDUCTOR statement and its parameters, see CONDUCTOR.

Examples

The following example illustrates an RVTV process.

```
$ RVTV statement
  RVTV rvtv1 {
        THICKNESS=0.0662 WMIN=4.0 SMIN=4.0
        CRT1=1.0e-03 CRT2=-7.0e-07
        DIFFUSION_TO_RVTV_SMIN=0.013
        }
```

See Also

- MULTIGATE
- RVTV Flow and Gate-All-Around FinFETs (GAAFETs) Extraction

Feedback

SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING

Models damage thickness variation of a conformal dielectric layer. Valid within a `DIELECTRIC` block for conformal dielectrics.

Syntax

```
SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING {
     SPACINGS {s1 s2 s3 ... sn}
     WIDTHS   {w1 w2 w3 ... wm}
     VALUES {
          v(s1,w1) v(s2,w1) ... v(sn,w1)
          v(s1,w2) v(s2,w2) ... v(sn,w2)
          …
          v(s1,wm) v(s2,wm) ... v(sn,wm)
     }
}
```

Arguments

Argument	Description
s1 s2 ...	Polygon to polygon spacing. The first value (s1) must equal the SMIN value. Units: microns
w1 w2 ...	Width of the polygon touching the conformal dielectric. The first value (w1) must equal the WMIN value. Units: microns
v(s1,w1) ...	Thickness of the dielectric damage layer for the corresponding spacing and width values Units: microns

Description

The `SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING` option models the thickness variation of side conformal layers. The width and spacing values are drawn dimensions, before any etch is applied.

This option should not be used for a planar dielectric layer that is not covertical with at least one conductor layer, because those planar layers contain bottom damage, not side damage.

Feedback

The following usage notes apply to the SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING option:

- The SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING option (variable dielectric thickness) is mutually exclusive with the DAMAGE_THICKNESS option (constant thickness) for a specific layer.
- If you use the SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING option, you must also use the DAMAGE_ER option to specify the dielectric constant of the damage layer.
- A conductor can have only one SW_T_VS_WIDTH_AND_SPACING or SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING option associated to it.

Examples

```
DIELECTRIC D3 {THICKNESS=0.12 ER=2.8 MEASURED_FROM=D2
  SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING {
     WIDTHS {0.05 0.07 0.1}
     SPACINGS {0.05 0.09 0.16 0.25}
     VALUES {0.010 0.009 0.008
             0.013 0.014 0.014
             0.016 0.018 0.020
             0.020 0.025 0.040  }
  }
  DAMAGE_ER=4.6
}
```

See Also

- DAMAGE_THICKNESS
- DAMAGE_ER
- SW_T_VS_WIDTH_AND_SPACING

Feedback

SIDE_TANGENT

Specifies a conductor or via sidewall angle in terms of the tangent of the angular shift from vertical. Valid within a `CONDUCTOR` or `VIA` block.

Syntax

```
SIDE_TANGENT = (coco_value, poco_value)
SIDE_TANGENT = tan_value
```

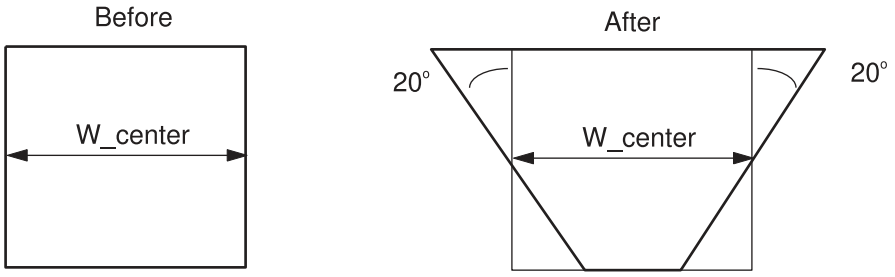
Arguments

Argument	Description
coco_value	Tangent of the sidewall angle in the direction of contact-to-contact (coco) spacing Units: none (ratio) Default: 0
poco_value	Tangent of the sidewall angle in the direction of gate-to-contact (poco) spacing Units: none (ratio) Default: 0
tan_value	Tangent of the sidewall angle in all directions Units: none (ratio) Default: 0

Description

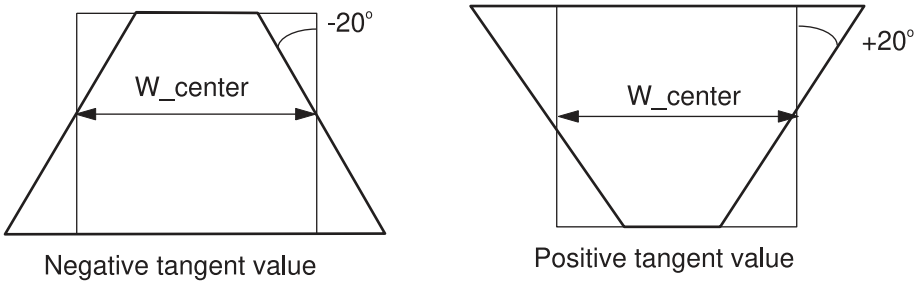
The `SIDE_TANGENT` option specifies the tangent of an angular shift from vertical of the sides of a feature, as shown in Figure 279.

Figure 279 Effect of the SIDE_TANGENT Option



Feedback

Figure 280 Positive and Negative Side Tangent Values



A positive tangent value results in a trapezoid with a top width larger than the center width, as shown in Figure 280. Similarly, a negative tangent results in a trapezoid with a top width smaller than the center width. If W is the width of the feature without any modification, and t is the layer thickness, the top and bottom dimensions change as follows:

W_top = W + (SIDE_TANGENT * t)

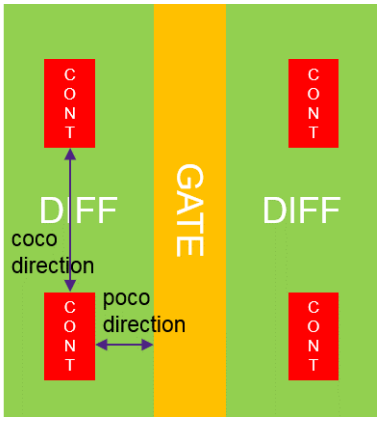
W_bottom = W - (SIDE_TANGENT * t)

You can specify the `SIDE_TANGENT` option with either one or two tangent values, as follows:

- A single tangent value represents the sidewall angle for both the x- and y-directions.
- Two tangent values represent separate sidewall angles for the direction between contacts (contact-to-contact or coco direction) and the direction between a contact and the polysilicon or gate shape (poly-to-contact or poco direction).

  You can specify two tangent values in either a `CONDUCTOR` or `VIA` block. In both cases, the `LAYER_TYPE` option must be set to `TALL_CONTACT`.

Figure 281 Directions for Side Tangent Values



Feedback

The following notes apply to the SIDE_TANGENT option in a CONDUCTOR block:

- The center width and cross-sectional area are not affected by the adjustment; therefore resistance is not affected. However, capacitance is affected due to the shape changes at the top and bottom surfaces.
- You can get more precise control by using the SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING or SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING options instead.
- All side tangent options are overridden by the ETCH_VS_WIDTH_AND_SPACING ETCH_FROM_TOP option, if it is used.

The following notes apply to the SIDE_TANGENT option in a VIA block:

- The option can only be applied to contact via layers, in other words, vias that connect a diffusion layer and a metal layer immediately above the gate layer.

Examples

The following example specifies a 20-degree angular shift (tan 20°=0.364):

```
CONDUCTOR met4 {
     SIDE_TANGENT=0.364 THICKNESS=0.66 WMIN=0.15
     SMIN=0.15 RPSQ=0.078
}
```

See Also

- ETCH_VS_WIDTH_AND_SPACING
- SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING
- SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING

Feedback

SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING

Specifies the side tangent as a function of the device type of the gate, width of the conductor, and spacing from neighboring polygons. Valid within a CONDUCTOR block.

Syntax

```
SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING {
  NUMBER_OF_TABLES = num_tables
  device_type_1 {
    CONTACT_TO_GATE_SPACINGS { s1 s2 s3 ... sn }
    WIDTHS { w1 w2 w3 ... wm }
    VALUES {
      v(s1,w1)  v(s2,w1)  ... v(sn,w1)
      v(s1,w2)  v(s2,w2)  ... v(sn,w2)
      ...
      v(s1,wm)  v(s2,wm)  ... v(sn,wm)   }
  }
  …
  device_type_n {
    CONTACT_TO_GATE_SPACINGS { s1 s2 s3 ... sn }
    WIDTHS { w1 w2 w3 ... wm }
    VALUES {
      v(s1,w1)  v(s2,w1)  ... v(sn,w1)
      v(s1,w2)  v(s2,w2)  ... v(sn,w2)
      ...
      v(s1,wm)  v(s2,wm)  ... v(sn,wm)   }
  }
}
```

Arguments

Argument	Description
num_tables	Number of tables
device_type	Device type name
s1 s2 s3	Spacing values Units: microns
w1 w2 w3	Width values Units: microns
v(s1,w1) v(S2,w1) …	Side tangent values Units: microns

Feedback

Description

The SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING table specifies the side tangent as a function of the device type of the gate, width of the conductor, and spacing from neighboring gate polygons.

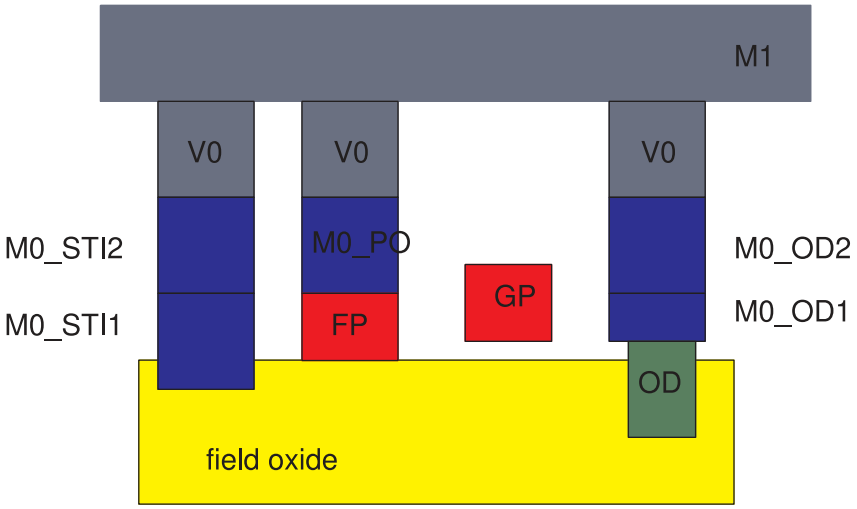
The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Examples

Figure 282 illustrates the difference between the two side tangent commands. If polygons on conductor layer M0_OD1 have a SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING table, the StarRC tool applies a side tangent only to the edges facing features on the gate polysilicon (GP) layer. The side tangent value depends on the device type of the gate, the width of the polygon on layer M0_OD1, and the spacing between the M0_OD1 polygon and the facing gate.

By contrast, if polygons on the other M0 layers such as M0_STI1 and M0_PO have a SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING table, the tool applies the side tangent to all four edges. The side tangent depends on the width of the polygon itself and the spacing between the polygon and neighbor polygons on layers with the same level.

Figure 282 Side Tangent as a Function of Width and Spacing



Feedback

The structure shown in Figure 282 is modeled by the ITF statements in the following example (the variable values v(sx,wy) would be replaced by values):

```
CONDUCTOR M0xxx {
  SIDE_TANGENT = st
  SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING {
    SPACINGS { 1.8 3.6 5.4 7.2 }
    WIDTHS   { 1.8000 2.7000 5.4000 10.8000 13.5000 }
    VALUES  {
      0.020 0.0230 0.0260 0.0290
      0.020 0.0230 0.0260 0.0290
      0.020 0.0230 0.0260 0.0290
      0.020 0.0230 0.0260 0.0290
      0.020 0.0230 0.0260 0.0290
    }
  }
  SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING {
    NUMBER_OF_TABLES=2
    G_1D5VIO_NMOS {
      CONTACT_TO_GATE_SPACINGS { s1 s2 s3 }
      WIDTHS { w1 w2 w3 }
      VALUES {
        v(s1,w1) v(s2,w1) v(s3,w1)
        v(s1,w2) v(s2,w2) v(s3,w2)
        v(s1,w3) v(s2,w3) v(s3,w3)
      }
    }
    G_CORE_NMOS {
      CONTACT_TO_GATE_SPACINGS { s1 s2 s3 }
      WIDTHS { w1 w2 w3 }
      VALUES {
        v(s1,w1) v(s2,w1) v(s3,w1)
        v(s1,w2) v(s2,w2) v(s3,w2)
        v(s1,w3) v(s2,w3) v(s3,w3)
      }
    }
  }
}
```

See Also

- SIDE_TANGENT
- SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING

Feedback

SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING

Specifies the side tangent as a function of the width of the conductor and spacing from neighboring polygons on layers with the same PIC levels as the conductor. Valid within a CONDUCTOR block.

Syntax

```
SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING {
    SPACINGS { s1 s2 s3 ... sn }
    WIDTHS   { w1 w2 w3 ... wm }
    VALUES {
      v(s1,w1)  v(s2,w1)  ... v(sn,w1)
      v(s1,w2)  v(s2,w2)  ... v(sn,w2)
      ...
      v(s1,wm)  v(s2,wm)  ... v(sn,wm)   }
  }
}
```

Arguments

Argument	Description
s1 s2 s3	Spacing values Units: microns
w1 w2 w3	Width values Units: microns
v(s1,w1) v(S2,w1) …	Side tangent values Units: microns

Description

The SIDE_TANGENT_VS_SI_WIDTH_AND_SPACING option specifies the side tangent as a function of the width of the conductor and spacing from neighboring polygons. Specify this table within a CONDUCTOR statement.

The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

See Also

- SIDE_TANGENT
- SIDE_TANGENT_VS_SI_WIDTH_AND_CCO_SPACING

Feedback

SMIN

Specifies the minimum spacing between two geometries. Valid within a CONDUCTOR or VIA block.

Syntax

```
SMIN = spacing_value
```

Arguments

Argument	Description
spacing_value	Minimum spacing value Units: microns

Description

The SMIN option specifies the minimum spacing between two features in a conductor or via layer. The SMIN value is the minimum layout dimension and should not be based on any processing steps (such as etches).

If you change the HALF_NODE_SCALE_FACTOR option, you must change the SMIN value accordingly. The StarRC tool does not modify the SMIN value automatically. For example, assume the SMIN value is 0.30 for a process in which the HALF_NODE_SCALE_FACTOR value is 1.0 (the default). If you set the HALF_NODE_SCALE_FACTOR value to 0.9, you should change the SMIN value to 0.27, which is calculated by applying the scale factor to the old SMIN value.

For via layers, the SMIN value is used only in transistor-level electromigration flows. The tool uses the SMIN value along with the WMIN value to calculate the maximum allowable spacing between vias for via merging. You can also provide these values by using the VIA_SMIN and VIA_WMIN commands in the StarRC command file. Values in the command file take precedence over values in the ITF file.

If the WMIN and SMIN statements are missing from the via layer definition in the ITF file and VIA_SMIN and VIA_WMIN are missing from the StarRC command file, default via merging occurs. If only one set of values is provided, the tool uses these values in the new flow. In the electromigration flow, the tool always performs via merging and ignores the MERGE_VIAS_IN_ARRAY command.

Examples

```
CONDUCTOR m1 {
     THICKNESS=1.00 WMIN=0.13 SMIN=0.15 RPSQ=0.015
}
```

Feedback

See Also

- HALF_NODE_SCALE_FACTOR
- VIA_SMIN
- VIA_WMIN
- WMIN
- Via Merging

Feedback

SPACER_ER_VS_WIDTH_AND_SPACING

Specifies relative permittivity values as a function of gate width and gate-to-contact spacing for tuning of coupling capacitance. Valid within a DIELECTRIC block for conformal dielectrics.

Syntax

```
SPACER_ER_VS_WIDTH_AND_SPACING {
    SPACINGS { s1 s2 … sn }
    WIDTHS   { w1 w2 … wm }
    VALUES   { er(s1,w1) er(s2,w1) … er(sn,w1)
               er(s1,w2) er(s2,w2) … er(sn,w2)
               …
               er(s1,wm) er(s2,wm) … er(sn,wm)   }
}
```

Arguments

Argument	Description
s1 s2 …	Gate to contact spacings specified in ascending order Units: microns
w1 w2 …	Gate widths specified in ascending order Units: microns
er(s1,w1) …	Relative permittivity for corresponding index values Units: none

Description

Specify the SPACER_ER_VS_WIDTH_AND_SPACING option within a DIELECTRIC block to vary the relative permittivity of conformal dielectrics. This option is most commonly used to modify coupling capacitance values rather than to model true physical effects.

The specified gate width and gate-to-contact spacing values are used as indexes for the two-dimensional table of relative permittivity values. Each combination of length *s* and width *w* has a corresponding relative permittivity value*er(s,w)*.

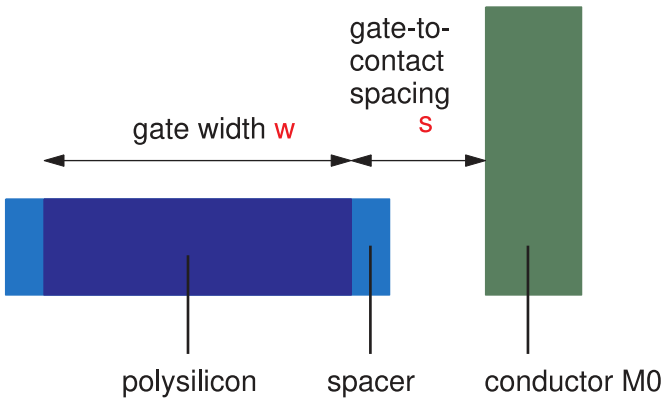
This keyword can only be used in a DIELECTRIC block that also include the following keywords:

- An ASSOCIATED_CONDUCTOR keyword specifying a conductor layer whose definition includes a LAYER_TYPE keyword set to a value of GATE
- An IS_CONFORMAL keyword

Feedback

Figure 283 shows the gate width and gate-to-contact spacing values required for the WIDTHS and SPACINGS lists.

Figure 283 Index Dimension Definitions



Examples

In the following example, when the spacing is 0.1 and the width is 0.4, the relative permittivity is 3.1. When the spacing is 0.1 and the width is 1.2, the relative permittivity is 2.8.

```
SPACER_ER_VS_WIDTH_AND_SPACING {
    SPACINGS { 0.1 0.2 }
    WIDTHS   { 0.4 1.0 1.2 }
    VALUES   { 3.1 3.2 3.4 3.5 2.8 2.6 }
}
```

See Also

- DIELECTRIC

Feedback

SW_T

Defines the sidewall thickness of a conformal dielectric layer. Valid within a DIELECTRIC block for conformal dielectrics.

Syntax

```
SW_T = swt_value
```

Arguments

Argument	Description
swt_value	Sidewall thickness Units: microns Default: dielectric THICKNESS value Valid values: 0 or a value greater than 0.005

Description

Use the SW_T option to specify a fixed thickness for the sidewall of a conformal dielectric layer.

A setting of SW_T=0 is allowed. If the SW_T value is not specified, the dielectric thickness is used for the sidewall thickness.

To model the sidewall thickness as a function of feature width and spacing, use the SW_T_VS_WIDTH_AND_SPACING option.

Sidewall Thickness of a Conformal Dielectric Layer on Gate Vias

You can use the SW_T statement to define one sidewall thickness for a conformal dielectric layer for each gate via, as shown in the following example:

```
DIELECTRIC vg_liner {
      ASSOCIATED_CONDUCTOR=viag
      ER=8
      IS_CONFORMAL
      SW_T=0.006
      THICKNESS=0
      TW_T=0
}
```

A gate via is a via connecting a poly layer, where LAYER_TYPE is GATE, FIED_POLY, or PODE_GATE, and a standard routing layer is specified without using LAYER_TYPE.

Feedback

Examples

```
DIELECTRIC D3 {
     THICKNESS = 0.2
     MEASURED_FROM = TOP_OF_CHIP
     SW_T = 0.15
     TW_T = 0.18
     ER = 5.9
}
```

See Also

- MEASURED_FROM
- THICKNESS
- TW_T
- SW_T_VS_WIDTH_AND_SPACING
- RVTV Flow and Gate-All-Around FinFETs (GAAFETs) Extraction

Feedback

SW_T_VS_WIDTH_AND_SPACING

Models sidewall thickness variation of a conformal dielectric layer. Valid within a `DIELECTRIC` block for conformal dielectrics.

Syntax

```
SW_T_VS_WIDTH_AND_SPACING {
     SPACINGS {s1 s2 s3 ... sn}
     WIDTHS   {w1 w2 w3 ... wm}
     VALUES {
          v(s1,w1) v(s2,w1) ... v(sn,w1)
          v(s1,w2) v(s2,w2) ... v(sn,w2)
          …
          v(s1,wm) v(s2,wm) ... v(sn,wm)
     }
}
```

Arguments

Argument	Description
s1 s2 ...	Polygon to polygon spacing. The first value (s1) must equal the SMIN value. Units: microns
w1 w2 ...	Width of the polygon touching the conformal dielectric. The first value (w1) must equal the WMIN value. Units: microns
v(s1,w1) ...	Thickness of the dielectric for the corresponding spacing and width values Units: microns

Description

The `SW_T_VS_WIDTH_AND_SPACING` option models the thickness variation of side conformal layers. The width and spacing values are drawn dimensions, before any etch is applied.

The following usage notes apply to the SW_T_VS_WIDTH_AND_SPACING option:

- The SW_T_VS_WIDTH_AND_SPACING option (variable dielectric thickness) is mutually exclusive with the SW_T option (constant thickness) for a specific layer.
- If you use the SW_T_VS_WIDTH_AND_SPACING option, the BW_T and TW_T options must be set to zero.
- A conductor can have only one associated dielectric (defined with the ASSOCIATED_CONDUCTOR option) with either an SW_T_VS_WIDTH_AND_SPACING or SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING option.

Examples

```
DIELECTRIC C3 {
   ER-4.6 IS_CONFORMAL TW_T=0
   ASSOCIATED_CONDUCTOR=M3
   SW_T_VS_WIDTH_AND_SPACING {
     WIDTHS {0.05 0.07 0.1}
     SPACINGS {0.05 0.09 0.16 0.25}
     VALUES {0.010 0.009 0.008
             0.013 0.014 0.014
             0.016 0.018 0.020
             0.020 0.025 0.040  }
   }
}
```

See Also

- BW_T
- TW_T
- SW_T
- SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING

Feedback

TALL_VIA_CONFIG

Specifies modeling of tall vias. Valid within a `VIA` definition.

Syntax

```
TALL_VIA_CONFIG {
   vname_1 {TOP {tx1 ty1 ... txn tyn} BOTTOM {bx1 by1 ... bxn byn}
   NOMINAL {nx1 ny1} RATIOS {r1 ... rn} PAIR_TO {va1 va2 ...}}

   vname_2 {TOP {tx1 ty1 ... txn tyn} BOTTOM {bx1 by1 ... bxn byn}
   NOMINAL {nx2 ny2} PAIR_TO {vb1 vb2 ...}}

   …
   vname_n {TOP {tx1 ty1 ... txn tyn} BOTTOM {bx1 by1 ... bxn byn}
   NOMINAL {nxn nyn} PAIR_TO {vn1 vn2 ...}}
}
```

Arguments

Argument	Description
vname_1 ... vname_n	A name for the via configuration
tx ty	X- and y-dimensions of the top of the via, in microns
bx by	X- and y-dimensions of the bottom of the via, in microns
nx ny	Nominal (drawn) x- and y-dimensions of the via, in microns
va1 va2 ...	Vias that might couple with vname_1
vb1 vb2 ...	Vias that might couple with vname_2
r1 ... rn	The thickness ratio of each via segments (Figure 285) from bottom to top is calculated by comparing with whole tall via. **Note:** The sum of `RATIOS` must be 1. However, if `RATIOS` is not specified, the thickness ratio of each segment is calculated by dividing 1 by the number of segments, as follows: $\frac{1}{\text{number of segments}}$ The {tx ty} and {bx by} pairs are defined in the same sequence as `RATIOS`. For example, • `TOP {tx1 ty1 tx2 ty2 tx3 ty3}` defines the top x/y dimensions of three via segments from bottom to top. • `BOTTOM {bx1 by1 bx2 by2 bx3 by3}` defines the bottom x/y dimensions of the three segments accordingly.

Feedback

Description

The TALL_VIA_CONFIG option models tall vias, which are via layers whose height is at least 20 times the thickness of the bottom conductor layer.

Tall vias connect two metal layers and have high aspect ratios, as shown in Figure 284. They might also have sloped sidewalls. To define tall vias, use the TALL_VIA_CONFIG option within a VIA block.

The TALL_VIA_CONFIG option contains a set of via configuration definitions, which are composed of the following components:

- A name for the via configuration
- X- and y-dimensions for the nominal or drawn (NOMINAL) via size, which identifies this via configuration in the layout. A nominal via size can only be used one time in a TALL_VIA_CONFIG block.

  All via sizes in the layout should be represented with a via configuration. Unmatched vias might have lower extraction accuracy and therefore cause the StarRC tool to issue a warning message.
- X- and y-dimensions for the top of the via (TOP) and the bottom of the via (BOTTOM). You can specify sloped sidewalls by making the top and bottom dimensions of the via different. The x-dimension and y-dimension can have different slopes.
- A list of named via configurations that might be capacitively coupled to this via configuration (PAIR_TO). This list usually includes the via configuration itself and other via configurations defined within the same TALL_VIA_CONFIG block. The list can also contain via configurations defined in other TALL_VIA_CONFIG blocks. It is not necessary to specify a pair of via configurations under both via names.

  Via configurations paired in this manner are analyzed with increased accuracy at the cost of increased time required to generate the nxtgrd file. Coupling capacitances that are not associated with a PAIR_TO keyword are analyzed using standard pattern-based extraction.

Figure 284 Tall vias

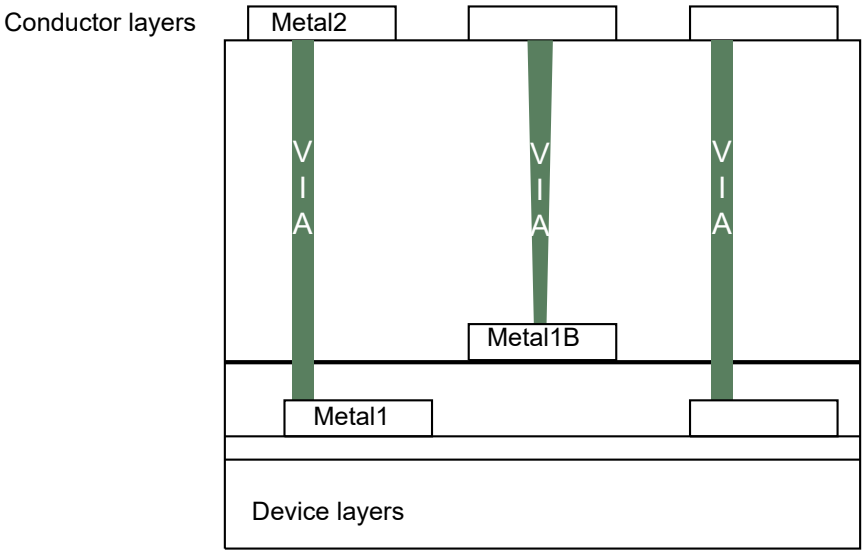
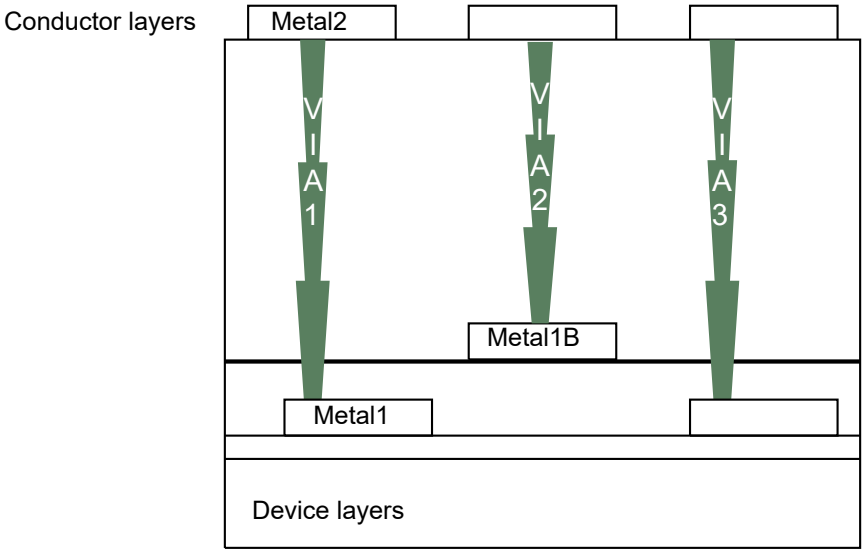


Figure 285 Tall vias with three segments



Examples

An example of the `TALL_VIA_CONFIG` option is as follows:

```
VIA Via1 {FROM=Metal1 TO=Metal2  RPV=5  AREA=0.01
   TALL_VIA_CONFIG {
      VIA1_Z {TOP {0.2 0.2 0.3 0.3 0.5 0.5}
      BOTTOM {0.1 0.1 0.2 0.2 0.4 0.4} NOMINAL {0.15 0.15}
      RATIOS {0.3 0.2 0.5 } PAIR_TO {VIA1_Z VIA1_X}}
```

```
        VIA1_A{TOP {0.2 0.2} BOTTOM {0.1 0.1} NOMINAL {0.15 0.15}
        PAIR_TO {VIA1_A VIA0_C}}

        VIA1_B{TOP {0.25 0.15} BOTTOM {0.1 0.07} NOMINAL {0.18 0.12}
        PAIR_TO {VIA1_B VIA1_X}}
        …
        VIA1_X{TOP {0.3 0.3} BOTTOM {0.3 0.3} NOMINAL {0.3 0.3}
        PAIR_TO {VIA1_X VIA1_B}}
}   }
```

See Also

- VIA
- Tall Via Modeling

Feedback

TC_ETCH_VS_WIDTH_AND_LENGTH

Specifies different etch values for the length and width sides of trench contact conducting layers. Valid within a CONDUCTOR block.

Syntax

```
TC_ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY PARALLEL_TO_GATE {
    LENGTHS { L1 L2 … Ln }
    WIDTHS  { w1 w2 … wm }
    VALUES  { e(w1,L1) e(w2,L1) ... e(wn,L1)
              e(w1,L2) v(w2,L2) ... e(wn,L2)
              ...
              e(w1,Lm) v(w2,Lm) ... e(wn,Lm)
    }
}
```

Arguments

Argument	Description
L1 L2 …	Trench lengths specified in ascending order Units: microns
w1 w2 …	Trench widths specified in ascending order Units: microns
e(w1,L1) ...	Etch values associated with the corresponding length and width indexes of trench contact conducting layers. This etch value to be applied in parallel to a gate direction. Units: microns Range: 0 to one half of the corresponding dimension

Description

Specify the TC_ETCH_VS_WIDTH_AND_LENGTH option within a CONDUCTOR statement to apply different trench etch values to the length and width sides of trench contact conducting layers. The etch values are applied only on the width of the trench contact conducting layer.

The following usage notes apply:

- The PARALLEL_TO_GATE keyword of the ETCH_VS_WIDTH_AND_SPACING command must be set with the TC_ETCH_VS_WIDTH_AND_LENGTH option to apply etch values only on the width of the conductor.
- The LAYER_TYPE keyword must be TRENCH_CONTACT.

- The trench length is the larger layout dimension and the trench width is the smaller layout dimension.
- The GATE_TO_CONTACT_SMIN values on the trench contact and field poly layers must be set up accurately to capture the applied etch effect on the coupling capacitance between the trench contact and field poly.
- The length and width values of the trench contact layers follow the orientation of the gate, where the length dimension is considered to be parallel to the gate direction and the width dimension is considered to be perpendicular to the gate direction.

The length and width values specified in the LENGTHS and WIDTHS keyword arguments are used as indexes for the table of etch values. Each combination of length *L* and width *w* has a corresponding etch value *e(w1, L1)*.

For example, consider a trench with length 0.2 microns and width 0.15 microns. The etch value is 0.005 microns. This etch value is applied in parallel to gate direction, so the length value remains the same. However, the width value is 0.14 (0.15 - 0.005 - 0.005).

Examples

In the following example, the StarRC tool sets the etch entry for length=0.045 and width=0.115 to (0.000, 0.000) because the parameter combination is invalid when the length is less than the width.

```
CONDUCTOR TC {
  THICKNESS=0.05 WMIN=0.025 SMIN=0.060 RPSQ=1.0
  GATE_TO_CONTACT_SMIN=0.02
  LAYER_TYPE=TRENCH_CONTACT
  TC_ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY PARALLEL_TO_GATE {
          LENGTHS { 0.045 0.115 0.200 }
          WIDTHS  { 0.045 0.115 }
          VALUES  { 0.015 0.002 0.003
                    0.000 0.015 0.005
                  }
  }
}
```

See Also

- CONDUCTOR
- LAYER_TYPE
- GATE_TO_CONTACT_SMIN
- ETCH_VS_WIDTH_AND_SPACING
- Trench Contact Process ITF Example
- ETCH_VS_WIDTH_AND_LENGTH

Feedback

TECHNOLOGY

Specifies the name of the process technology for tracking and identification purposes.

Syntax

```
TECHNOLOGY = process_name
```

Arguments

Argument	Description
process_name	The process name represented by a single word that can contain alphanumeric characters and underscores

Description

The `TECHNOLOGY` statement is mandatory and should precede all other statements, but it does not need to be the first line of the ITF file.

The output of the grdgenxo tool is stored in the *process_name*.nxtgrd file.

Examples

In the following example, the process name is example_tech, and the grdgenxo tool output is stored in the example_tech nxtgrd file:

```
TECHNOLOGY = example_tech
```

Feedback

THICKNESS

Specifies the thickness of a dielectric or conductor layer. Valid within a within a `CONDUCTOR` or `DIELECTRIC` block.

Syntax

```
THICKNESS = layer_thk
```

Arguments

Argument	Description
layer_thk	The thickness of the dielectric or conductor Units: microns Range: 0.001 or larger

Description

The dielectric or conductor thickness measured from the top of the dielectric layer below it. The reference point can be changed by setting a value for the `MEASURED_FROM` option. Specifying `THICKNESS=0` is allowed only for a conformal dielectric layer; for planar layers, the thickness value should not be 0.

Examples

```
CONDUCTOR M2 { THICKNESS=0.60 WMIN=0.55 SMIN=0.50 RPSQ=0.062 }
DIELECTRIC D2 { THICKNESS=1.20 ER=3.9 }
```

See Also

- MEASURED_FROM
- THICKNESS
- TW_T

Feedback

THICKNESS_VARIATION_VS_MASK

Defines conductor thickness variation based on the mask number. Valid in a `CONDUCTOR` block for transistor-level flows.

Syntax

```
THICKNESS_VARIATION_VS_MASK {
           (m1, tv1)
           (m2, tv2)
           ...
           (mn, tvn)
}
```

Arguments

Argument	Description
m1, m2, ...	Mask ID
tv1, tv2, ...	Thickness variation for the corresponding mask ID, expressed as the ratio to the nominal thickness Units: none Range: -0.4 to 0.4, inclusive

Description

The `THICKNESS_VARIATION_VS_MASK` command specifies conductor thickness as a function of mask ID. This command allows you to model a design in which a conductor layer has different thicknesses in different parts of the circuit, especially used in conjunction with the `silicon_marker_layers` command in the mapping file.

The following usage notes apply:

- The `THICKNESS_VARIATION_VS_MASK` command specifies a layer thickness relative to another layer thickness. The two conductor layers must be linked by reciprocal `ASSOCIATED_CONDUCTOR` statements.
- The bottom heights of the two associated conductor layers must be identical. By definition, the two layers are covertical, which means they overlap in the vertical dimension.
- Interlayer dielectric (ILD) variations are not allowed for either of the two associated conductors, because ILD variations cause the conductor bottom heights to be different.

- You can use different thickness variation tables for the two associated conductors.
- The `THICKNESS_VARIATION_VS_MASK` table must be the only thickness variation specification for that layer.

You can use a silicon marker layer to change the mapping of one database conductor layer to another database conductor layer, if the two layers are associated by using the `ASSOCIATED_CONDUCTOR` keyword in their respective `CONDUCTOR` blocks. Specify the new layer by using the `silicon_marker_layers` command in the mapping file.

For example, the following mapping file line assigns the portion of layer M0 that overlaps marker layer M0_logic1 to layer M0_thin:

```
silicon_marker_layers
   M0_logic1    M0   ITF = M0_thin
```

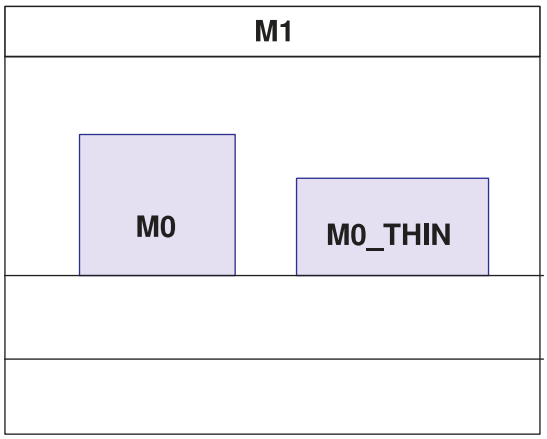
You can also use a silicon marker layer to specify a mask number that affects the layer properties through other ITF commands. The following line remaps the portion of layer M0 that overlaps marker layer M0_memory to layer M0_thick. In addition, a mask ID of 5 is used in any ITF command that specifies a mask-dependent thickness variation command.

```
silicon_marker_layers
   M0_memory    M0   ITF = M0_thick   MASK = 5
```

Examples

The following example is for a process with two covertical conductor layers, M0 and M0_THIN, as shown in Figure 286. Conductor layers M0 and M0_THIN are associated with each other, but they have different device types and different methods of modeling thickness variation.

Figure 286 Conductor Layer With Two Thicknesses



The ITF file contains the following lines:

```
CONDUCTOR M0 {
      THICKNESS=0.6 WMIN=0.34 SMIN=0.40
      ASSOCIATED_CONDUCTOR = M0_THIN
      POLYNOMIAL_BASED_THICKNESS_VARIATION { ... }
      ETCH_VS_WIDTH_AND_SPACING {
         NUMBER_OF_MASKS = 6
         MASKS(2,3) {...}
      ...
      }
      DEVICE_TYPE {PMOS_1 NMOS_1}  $Devices associated with layer M0
}
CONDUCTOR M0_THIN {
      THICKNESS=0.8 WMIN=0.50 SMIN=0.40
      ASSOCIATED_CONDUCTOR = M0
      ETCH_VS_WIDTH_AND_SPACING {
         NUMBER_OF_MASKS = 6
         MASKS(2,3) {...}
      }
      THICKNESS_VARIATION_VS_MASK {
         (3,0.05) (4,0.05) (5,0.07) (6,0.07) ...
      }
      DEVICE_TYPE {PMOS_2 NMOS_2}  $Devices associated with layer M0_THIN
}
```

Feedback

See Also

- ETCH_VS_WIDTH_AND_SPACING
- RPSQ
- RPSQ_VS_WIDTH_AND_SPACING
- silicon_marker_layers

Feedback

THICKNESS_VS_DENSITY

Models the thickness of a conductor as a function of its feature density. Valid within a CONDUCTOR block.

Syntax

```
THICKNESS_VS_DENSITY [ RESISTIVE_ONLY | CAPACITIVE_ONLY ]
  {(d1 r1) (d2 r2)  … }
```

Arguments

Argument	Description
RESISTIVE_ONLY	Applies thickness adjustment to resistance only
CAPACITIVE_ONLY	Applies thickness adjustment to capacitance only
d1, d2, …	Density value (fraction of area occupied by features) Units: none
r1, r2, …	Relative change in thickness (dT/T ratio). A negative value indicates a thickness decrease; a positive value indicates a thickness increase. Units: none Range: -1 to 1

Description

In chemical-mechanical polishing (CMP) processes, the density of features in the vicinity of a specific feature affects the amount of polishing in that area. The THICKNESS_VS_DENSITY option provides a method of adjusting conductor thickness based on local feature density.

Thickness variation affects both resistance and capacitance. However, you can use different coefficients for resistance and capacitance calculations by using the RESISTIVE_ONLY or CAPACITIVE_ONLY options.

Analyze conductor thickness as a function of the feature density as follows:

- Use the DENSITY_BOX_WEIGHTING_FACTOR option to specify the size and weighting factors for up to five density boxes (evaluation regions), which are square boxes centered around the conductor of interest.

  If you do not specify the DENSITY_BOX_WEIGHTING_FACTOR option, the default is a single density box with a size of 50 µm and a weighting factor of 1.

- Use the THICKNESS_VS_DENSITY option to specify density values and the thickness variations that correspond to those densities.

The tool performs the analysis as follows:

1. Based on the layout, determine the conductor feature density within a density box
2. Scale the density by the density box weighting factor
3. Repeat steps 1 and 2 for each supplied density box
4. Determine the effective density by adding the scaled density values
5. Look up the thickness variation for the effective density
6. Multiply the thickness variation to the nominal thickness to calculate the adjusted thickness
7. Calculate resistance and capacitance based on the adjusted conductor thickness

Examples

An example of the single-box method is as follows:

```
CONDUCTOR metal3 {
     THICKNESS=0.5 SMIN=0.25 WMIN=0.25 RPSQ=0.01
     THICKNESS_VS_DENSITY RESISTIVE_ONLY
          {(0.1,0.3) (0.2,0.2)(0.3,0.1)(0.4,-0.1)}
}
```

An example of the multiple-box method is as follows:

```
CONDUCTOR metal3 {
     THICKNESS = 0.5 SMIN = 0.25 WMIN=0.25 RPSQ = 0.01
     THICKNESS_VS_DENSITY{(0.1 0.3)(0.2 0.2)(0.3 0.1)(0.4 -0.1)}
     DENSITY_BOX_WEIGHTING_FACTOR {
          (10 1) (20 0.23) (30 0.29)
          (40 0.18) (50 -0.12)
     }
}
```

Feedback

See Also

- DENSITY_BOX_WEIGHTING_FACTOR
- THICKNESS_VS_WIDTH_AND_SPACING
- Conductor Layer Thickness Variation

Feedback

THICKNESS_VS_DENSITY_AND_WIDTH

Thickness variation with respect to density and width. Valid within a CONDUCTOR block.

Syntax

```
THICKNESS_VS_DENSITY_AND_WIDTH  {
     [SI_]WIDTHS {w1 w2 ... wn}
     [SI_]DENSITIES  {d1 d2 ... dm}
     VALUES  { v(w1,d1) v(w2,d1) ... v(wn,d1)
               v(w1,d2) v(w2,d2) ... v(wn,d2)
               ...
               v(w1,dm) v(w2,dm) ... v(wn,dm)  }
}
```

Arguments

Argument	Description
w1 w2 … wn	Width of the conductor Units: microns
d1 d2 … dm	Metal density in the density box Valid values: 0.0 to 1.0 Units: microns
v(w1,d1) v(w2,d1) …	Relative thickness change values (deltaT/T ratio) Units: none

Description

Specify this option to model conductor thickness variation with respect to feature density and feature width using a two-dimensional lookup table.

By default, feature density is calculated within a square box 50 microns on a side, centered on the conductor of interest. You can change the size of this box by using the DENSITY_BOX_WEIGHTING_FACTOR option.

The following ITF options provide alternative methods for modeling conductor thickness variation. Therefore, they cannot be specified for the same conducting layer when the THICKNESS_VS_DENSITY_AND_WIDTH option is used.

- THICKNESS_VS_DENSITY
- THICKNESS_VS_WIDTH_AND_SPACING
- POLYNOMIAL_BASED_THICKNESS_VARIATION

Feedback

Specifying As-Drawn or Silicon Values for Width and Density

Whether the width and density values use as-drawn or post-etch (silicon) values is determined by the presence or absence of the `SI_` prefix in the `WIDTHS` and `DENSITIES` keywords.

The global `USE_SI_DENSITY` command in the ITF file might conflict with these keywords. Table 104 describes the allowed usages.

Table 104 Effect of USE_SI_DENSITY Settings

THICKNESS_VS_DENSITY_AND_WIDTH Keyword	USE_SI_DENSITY: YES	USE_SI_DENSITY: NO	USE_SI_DENSITY not used
WIDTHS	drawn width	drawn width	drawn width
DENSITIES	silicon density	drawn density	drawn density
SI_WIDTHS	error	error	silicon width
SI_DENSITIES	error	error	silicon density

Limits

When the input exceeds the maximum value in the `[SI_]WIDTHS` or `[SI_]DENSITIES` specifications, the input used in the table lookup is the same as for the maximum value. For example, if an input w is larger than wn, wn is used. Similarly, if an input w is smaller than w1, w1 is used.

Examples

```
CONDUCTOR m1 {
   THICKNESS=0.4
      THICKNESS_VS_DENSITY_AND_WIDTH {
        SI_WIDTHS { 0.2 0.4 0.8 2.0 }
        SI_DENSITIES { 0.1 0.5 0.8 }
        VALUES {
          0.1 0.0 -0.1 -0.3
         -0.2 0.1 0.2 0.3
         -0.3 -0.1 0.0 0.1  }
```

See Also

- DENSITY_BOX_WEIGHTING_FACTOR
- THICKNESS_VS_DENSITY
- THICKNESS_VS_WIDTH_AND_SPACING
- POLYNOMIAL_BASED_THICKNESS_VARIATION

Feedback

THICKNESS_VS_SPACING

Specifies planar dielectric thickness spacing in-between the shapes of the same conductor. Valid within a `DIELECTRIC` block.

Syntax

```
THICKNESS_VS_SPACING {
      SPACINGS {s1, s2, ... ,sn}
      VALUES {t(s1), t(s2), ..., t(sn)}
    }
```

Arguments

Argument	Description
s1, s2, …	Spacing between two conductors, in ascending order Units: microns
t(s1), t(s2), …	Thickness of the modified planar layer

Description

The `THICKNESS_VS_SPACING` ITF statement modifies the thickness of the planar dielectric layer (L_d) that includes the ITF table data. The associated conductor layer (L_{cond}) is the last conductor layer defined before the $L_{d.}$ layer. The $L_{d.}$ layer can be any single planar dielectric layer (but not the conformal layer) that is defined between the L_{cond} layer and the next back-end-of-line conductor layer.

The tool generates dielectric thickness models based on the following conditions:

- L_{cond} layer is not associated with a side-wall of the conformal layer

  If the L_{cond} layer is not associated with a side-wall of the conformal layer with the data either from the `SW_T_VS_WIDTH_AND_SPACING` or `SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING` table, then the tool generates the dielectric thickness models based on the spacing points defined with the `THICKNESS_VS_SPACING` statement.

- L_{cond} layer is associated at the same time with a side-wall of the conformal layer

  If the L_{cond} layer is associated at the same time with a side-wall of the conformal layer with the data either from the `SW_T_VS_WIDTH_AND_SPACING` or `SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING` table, then the tool while generating dielectric thickness models identifies the presence of another set of spacing points that are dependent table entries for the same conductor layer.

For example, if the name of spacing entries related to the conformal table is `Ssw{}` and the spacing entries related to a new dielectric thickness is `Sdt{}`, the grdgenxo tool merges the two spacing array points to form the new spacing array. A minimum dimension threshold is used to decide when two spacing entries are the same. The new spacing array is the union of spacing points, $S\{\} = Ssw \cup Sdt$. The width points are from the `SW_T_VS_WIDTH_AND_SPACING` or the `SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING` table.

For dielectric thickness,

- `Ssw`: Indicates that the data is either from the `SW_T_VS_WIDTH_AND_SPACING` or `SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING` table.
- `Sdt`: Indicates that the data is from the `THICKNESS_VS_SPACING` table.

Modifying Thickness Value for Successive Planar Dielectrics

The following figure displays the thickness variation (dt) caused by the `THICKNESS_VS_SPACING` table, localized between two conductor instances. This feature applies layer-wise thickness variation.

The specified dt values lead to the following variation of thickness:

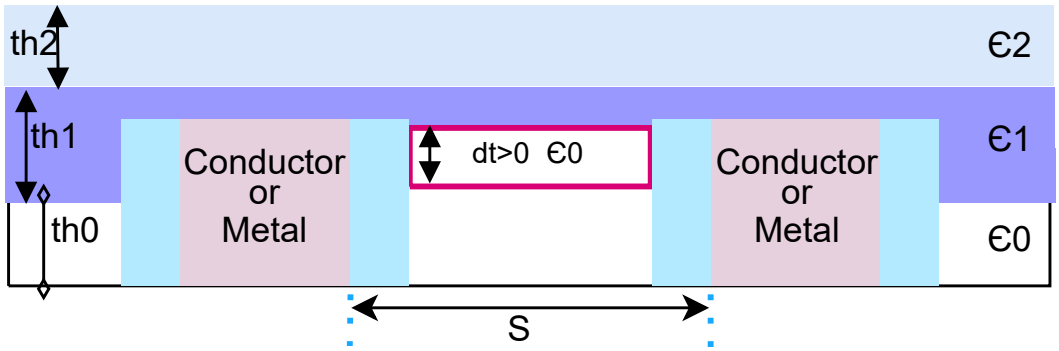
- Increase in thickness, dt > 0:

  Induces overlapping between the affected planar dielectric layer (L0) and the next-upper planar layer (L1).

  The epsilon (ε) of layer L0 takes precedence over the next-upper planar layer (L1) (figure). Hence, L0 thickness increases by dt and L1 must be decreased by the same dt for the inter-metal distances at the original value.

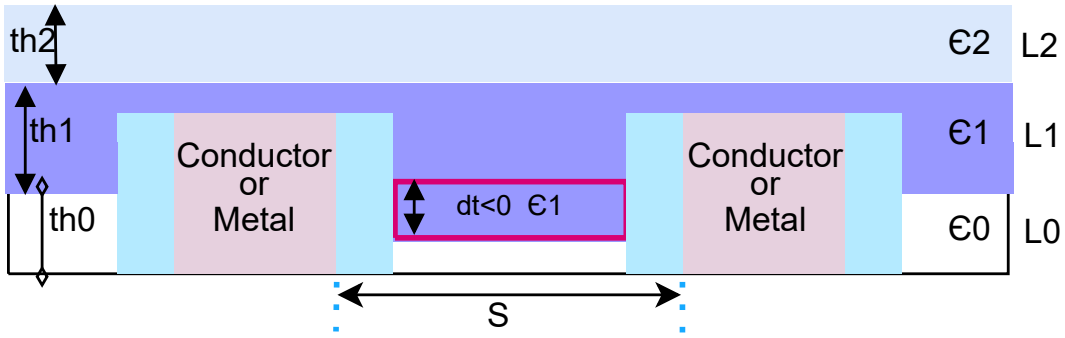
  In the following figure

  - dt indicates thickenss variation.
  - Epsilon (ε) indicates epsilon of a layer.
  - Ssw indicates dielectric conformal table.
  - Sdt indicates dielectric thickness.



- Decrease in thickness, dt < 0

  Introduces void between the two planar layers (L0 and L1). The epsilon of this area is filled with the epsilon of the next-upper planar layer (L1).

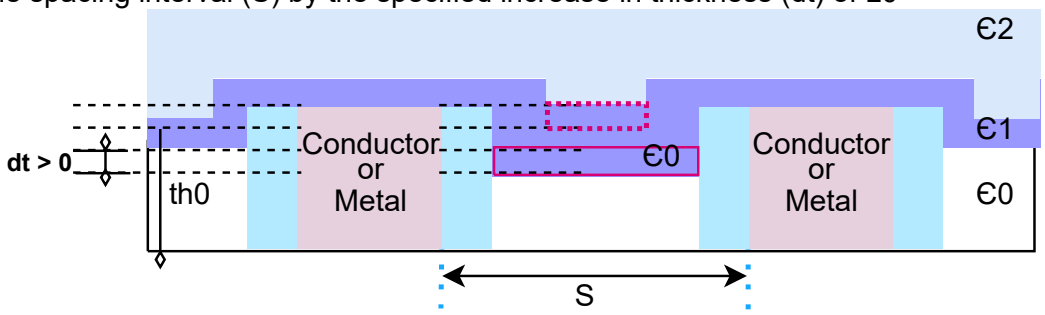


Modifying Thickness Value for a Top Conformal Dielectrics

Top conformal indicates top-of-chip dielectrics. The specified dt values lead to the following variation of thickness:

- Increase in thickness of planar dielectric layer (L0), dt > 0

  Maintains the conformal layer (L1) with the same nominal thickness across the horizontal line. The next-dielectric layer (L2) that should get planarized is thinned across the spacing interval (S) by the specified increase in thickness (dt) of L0

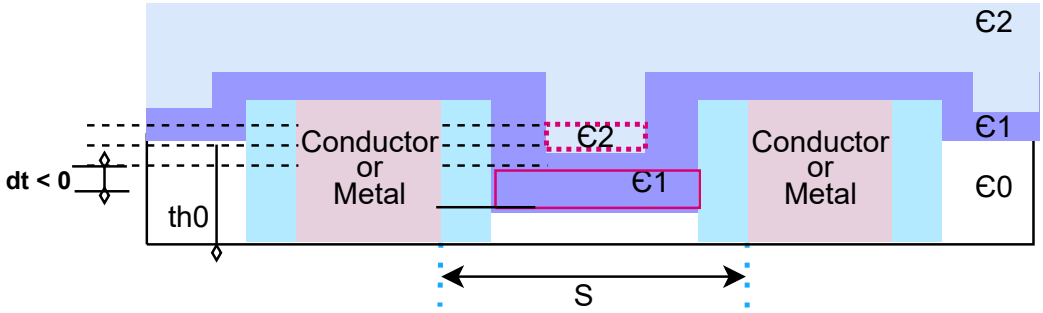


(figure).

Feedback

- Decrease in thickness of planar dielectric layer (L0), dt < 0

  Maintains the conformal layer (L1) with the same nominal thickness across the horizontal line. The next-dielectric layer (L2) that should get planarized is thickened across the spacing interval (S) by the specified increase in thickness (dt) of L0 (figure).



Rules When Applying THICKNESS_VS_SPACING

To apply thickness value using the `THICKNESS_VS_SPACING` statement,

- The CONDUCTOR layer should have no special `LAYER_TYPE`, including the device region.
- Only one layer with the `THICKNESS_VS_SPACING` table is allowed between the two successive `CONDUCTOR` layers.
- If the thickness value of a table exceeds the nominal thickness value, the resulting positive thickness difference cannot exceed the thickness value of the next upper Layer (L1).
- At least one planar dielectric (L1) must be available that is not declared as `MEASURED_FROM=TOP_OF_CHIP` between the modified planar dielectric (L0) and the next upper conductor layer. This layer compensates for the thickness change, so you do not see any change in the neighboring inter-metal distances.
- For all tables, allows only interpolation and no extrapolation is allowed outside spacing bounds.
- For all spacings beyond the upper-specified layer (sn) or for all single conductors with no neighbors, the thickness value is that of the last element in the table. This thickness value coincides with that of the nominal thickness of the planar dielectric layer (L0).

See Also

- THICKNESS_VS_WIDTH_AND_SPACING
- SW_T_VS_WIDTH_AND_SPACING

Feedback

- SIDE_DAMAGE_THICKNESS_VS_WIDTH_AND_SPACING
- DIELECTRIC

Feedback

THICKNESS_VS_WIDTH_AND_SPACING

Models the thickness of a conductor as a function of its width and spacing. Valid within a `CONDUCTOR` block.

Syntax

```
THICKNESS_VS_WIDTH_AND_SPACING [RESISTIVE_ONLY | CAPACITIVE_ONLY] {
    SPACINGS { s1 s2 … sn }
    WIDTHS   { w1 w2 … wm }
    VALUES   { v(s1,w1) v(s2,w1) … v(sn,w1)
               v(s1,w2) v(s2,w2) … v(sn,w2)
               …
               v(s1,wm) v(s2,wm) … v(sn,wm) }
}
```

Arguments

Argument	Description
`RESISTIVE_ONLY`	Applies thickness adjustment to resistance only
`CAPACITIVE_ONLY`	Applies thickness adjustment to capacitance only
`s1 s2 …`	Conductor spacings specified in ascending order Units: microns
`w1 w2 …`	Conductor widths specified in ascending order Units: microns
`v(s1,w1) …`	Relative thickness change (dT/T ratio) for corresponding index values Units: none

Description

In this method, the variation of thickness is modeled as a function of the width of a conductor and relative spacing to the neighboring conductor.

The effective thickness is calculated as follows:

$$T = Tnom \times (1 + RT(Deff) + RT(W, S) + RT(SiW))$$

In this equation,

- T_{nom} is the nominal thickness specified in the ITF file.
- RT(Deff) is the relative thickness change due to density.

- RT(W,S) is the relative change in thickness due to width and spacing.
- RT(SiW) is the relative change in thickness due to silicon width.

Examples

```
CONDUCTOR m1 {
     THICKNESS = 0.60 WMIN 0.25 SMIN = 0.25
     THICKNESS_VS_WIDTH_AND_SPACING {
          SPACINGS { 0.25 0.30 0.50}
          WIDTHS {0.25 0.4 0.50}
          VALUES {0.3 0.2 0.1
                  0 -0.1 -0.20
                  -0.3 -0.2 -0.1}
     }
}
```

See Also

- DENSITY_BOX_WEIGHTING_FACTOR
- THICKNESS_VS_DENSITY

Feedback

TO

Specifies a conductor layer or multiple conductor layers connected to a via. Valid within a `VIA` block.

Syntax

```
TO = layer | TO {layer [layer2...]}
```

Arguments

Argument	Description
layer	Conductor layer or multiple conductor layers connected by the via

Description

The `TO` option specifies the upper or lower layer connected by the via. A via can connect only two conductor layers. The tool considers both the `TO = layer` and `TO {layer...}` options specified for the `CONDUCTOR` layer

If you specify multiple layers with the `TO` option, all layers must have the same top height if they are lower layers or the same bottom height if they are upper layers. All layers must have the same LAYER_TYPE if you specify the layer type.

Examples

The following example shows how to specify a layer with the `TO = layer` option:

```
VIA v1 {
      FROM = m2
      TO = m3
      RPV = 40
      AREA = 0.16
}
```

The following example shows how to specify a layer with the `TO {layer...}` option:

```
VIA v2 {
  FROM {diff}
  TO {tc1 tc2}
  AREA = 0.25
  RPV = 5
}
```

See Also

- FROM

Feedback

TRENCH_CONTACT_EXTENSION

Specifies an extension of the trench contact layer for resistance extraction. Valid within a `CONDUCTOR` block.

Syntax

```
TRENCH_CONTACT_EXTENSION = extension_value
```

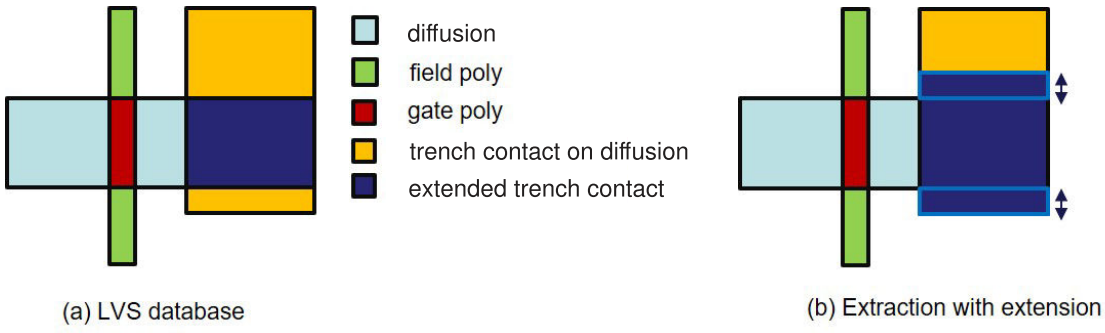
Arguments

Argument	Description
extension_value	Trench contact extension Units: microns

Description

The `TRENCH_CONTACT_EXTENSION` option can only be used in a conductor definition block that contains a `LAYER_TYPE` option set to `TRENCH_CONTACT`. The trench contact extension is the continuation region between the trench contact on the diffusion layer and the extended trench contact. The resistance of the extension region is extracted using the resistivity of the trench contact over diffusion layer. Figure 287 illustrates the extraction region.

Figure 287 Trench Contact Extension



See Also

- LAYER_TYPE

Feedback

TSV

Describes a through-silicon via (TSV) layer.

Syntax

```
TSV tsv_name {
      FROM = layer1
      TO = layer2
      RHO = rho_value
      AREA = area_value
      THICKNESS = thickness_value
      INSULATION_THICKNESS = ins_thickness_value
      INSULATION_ER = er_value
      [CRT1 = lin_coeff] [CRT2 = quad_coeff] [T0 = nominal_temp]
      CSUB_VS_SPACING { (s1,c1) (s2, c2) … (sn,cn)}
      RSUB_VS_SPACING { (s1,r1) (s2, r2) … (sn,rn)}
      CEFF_VS_FREQUENCY_AND_SPACING {
      SPACINGS {s1 s2 s3 … }
      FREQUENCY { f1 f2 f3 … }
      VALUES { v(s1 f1) v(s2 f1) v(s3 f1)…
               v(s2 f1) v(s2 f2) v(s2 f3) … } }
}
```

Arguments

Argument	Description
layer1	Conductor layer connected by the TSV
layer2	Conductor layer connected by the TSV
rho_value	Resistivity of the TSV Units: ohm-microns
area_value	Area of the TSV Units: square microns
thickness_value	Thickness of the TSV Units: microns
ins_thickness_value	Thickness of the insulation layer between the TSV and the substrate Units: microns
er_value	Relative permittivity of the TSV insulation layer
lin_coeff	Layer-specific linear temperature coefficient Default: 0

Feedback

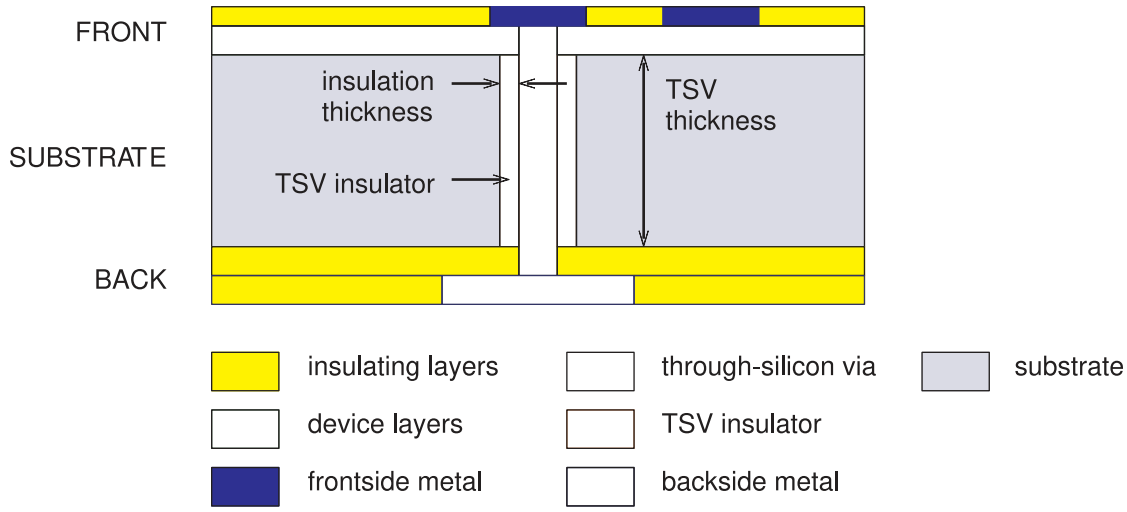
Argument	Description
quad_coeff	Layer-specific quadratic temperature coefficient Default: 0
nominal_temp	Nominal temperature for the layer Units: degrees Celsius Default: temperature specified by GLOBAL_TEMPERATURE
CSUB_VS_SPACING (s_n,c_n)	The distance and substrate-related capacitance between one through-silicon via and its neighboring through-silicon via You can specify a maximum of eight spacing-capacitance pairs. Units: microns
RSUB_VS_SPACING (s_n,r_n)	The distance and substrate-related resistance between one through-silicon via and its neighboring through-silicon via. You can specify a maximum of eight spacing-resistance pairs. Units: microns
CEFF_VS_FREQUENCY_AND_SPACING	The effective capacitance between individual through-silicon vias. The CEFF_VS_FREQUENCY_AND_SPACING table is provided by the foundry. The effective capacitance is modeled by distance between TSV pairs and the working frequency in simulation. Use the netlist for SPICE and static timing analysis tools.
f1 f2 ...	The operating frequencies Units: Hertz
s1 s2 ...	The spacing between through-silicon vias Units: microns
v(s1,f1) v(s1,f2) ...	The effective capacitance between through-silicon vias The numbers in the VALUES field are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Description

A through-silicon via (TSV) connects a conductor layer on the frontside of the substrate to a conductor layer on the backside, extending all the way through the substrate as shown in Figure 288. The via is surrounded by an insulation layer to separate it from the substrate.

Feedback

Figure 288 Cross Section of Through-Silicon Via



To model a through-silicon via, you must include descriptions of the frontside and backside layers in the ITF file. The specification of the frontside and backside layers follows standard ITF syntax to describe the layers in order from the outside in. In other words, describe the backside layers as if the substrate is on the bottom and the layers are on the top, even though in reality the configuration is reversed.

Note:

The `TSV` command cannot be used with simultaneous multicorner extraction.

Place the `TSV` statement after the frontside ITF statements and before the backside ITF statements. For an example of an ITF file that describes a through-silicon via, see Through-Silicon Via ITF Example.

For high-frequency designs, you should consider the effects of coupling of individual through-silicon vias through the substrate. Two approaches are available:

- For SPICE flows, you can generate a full RC network, as shown in Figure 289, using the `CSUB_VS_SPACING` and `RSUB_VS_SPACING` tables provided by the foundry.
- For static timing analysis flows, you might prefer to generate a netlist that includes only capacitances between individual vias, because some analysis tools cannot accept parallel RC networks. In this case, use the `CEFF_VS_FREQUENCY_AND_SPACING` table provided by the foundry. This model uses an effective capacitance value between individual through-silicon vias, as shown in Figure 290.

For all tables, if the frequency or spacing is outside the table boundaries, the StarRC tool uses the parameter value at the boundary. Within the table boundaries, the tool uses linear interpolation to determine the parameter value.

Feedback

Figure 289 RC Model for SPICE Flows

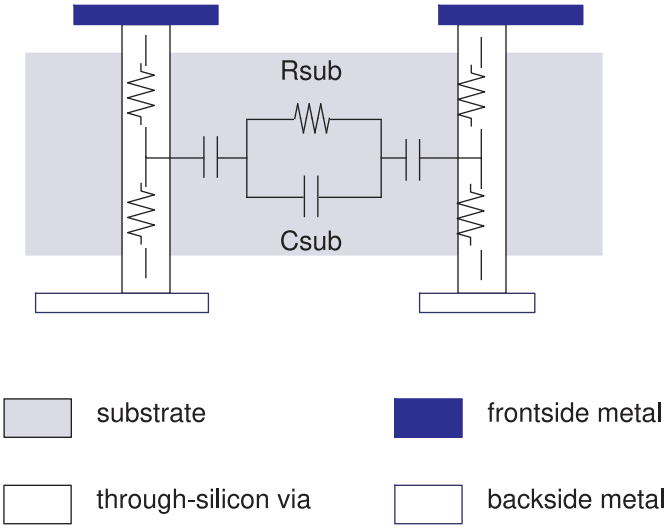
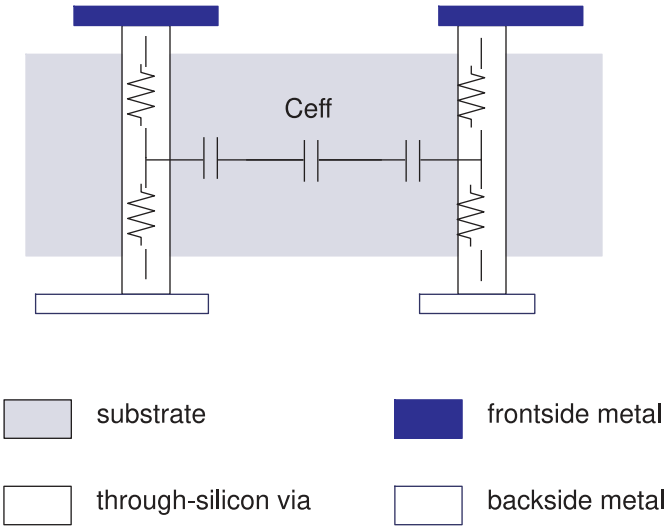


Figure 290 Effective Capacitance Model for Static Timing Analysis



Feedback

Examples

The following example illustrates a through-silicon via definition. The frontside layers are defined before the `TSV` statement and the backside layers are defined after it.

```
TECHNOLOGY = tsv_process
GLOBAL_TEMPERATURE = 25.0

$ Frontside ITF statements
CONDUCTOR M1 {
  THICKNESS=0.3 WMIN=0.12 SMIN=0.12 SIDE_TANGENT=0.2
  CRT1=7.0e-03 CRT2=-8.0e-07
}
DIELECTRIC ILD_B { ER=5.5 THICKNESS=0.5 }
DIELECTRIC GATE_OXIDE { ER=4.3 THICKNESS=0.03 }
DIELECTRIC FOXIDE_A { ER=5.0 THICKNESS=0.1 }
CONDUCTOR DIFFUSION {
  THICKNESS=0.1 WMIN=0.1 SMIN=0.06
  CRT1=8.0e-03 CRT2=-1.0e-07 RPSQ=36.0
}
DIELECTRIC FOXIDE { ER=5.0 THICKNESS=0.2 }
VIA via1 { FROM=M1 TO=M2 AREA=0.03 RPV=2.5 CRT1=3.0e-04 CRT2=-6.0e-06 }

$ TSV statement
TSV tsv {
  FROM=M1 TO=M1b RHO=0.05 AREA=49.0 THICKNESS=20.0
  INSULATION_THICKNESS = 0.4 INSULATION_ER = 5.5
  CRT1=7.0e-03 CRT2=-3.0e-08
}

$ Backside ITF statements
DIELECTRIC PASS { ER=9.0 THICKNESS=2.0 }
DIELECTRIC DIELb { ER=5.0 THICKNESS=1.0 }
CONDUCTOR M1b {
  THICKNESS=2.0 WMIN=4.0 SMIN=4.0 SIDE_TANGENT=-0.2
  CRT1=1.0e-03 CRT2=-7.0e-07
}
DIELECTRIC ILDB{ ER=5.2 THICKNESS=1.0 }
```

See Also

- 3D_IC
- 3D_IC_FILTER_DEVICE
- 3D_IC_FLOATING_SUBSTRATE
- 3D_IC_SUBCKT_FILE
- 3D_IC_TSV_COUPLING_EXTRACTION
- Through-Silicon Vias

Feedback

TW_T

Defines the topwall thickness of a conformal layer. Valid within a DIELECTRIC block for conformal dielectrics.

Syntax

```
TW_T = thickness
```

Arguments

Argument	Description
thickness	Topwall thickness Units: microns Default: dielectric THICKNESS value Valid values: 0 or a value greater than 0.005

Description

TW_T=0 is allowed for conformal dielectrics. If the TW_T option is not specified, the dielectric thickness is used for the topwall thickness.

Examples

```
DIELECTRIC D3 {
     THICKNESS=0.2  MEASURED_FROM=TOP_OF_CHIP
     SW_T=0.15 TW_T=0.18 ER=5.9
}
```

See Also

- MEASURED_FROM
- SW_T
- THICKNESS

Feedback

USE_SI_DENSITY

Specifies whether to use density based on silicon (post-etch) or drawn dimensions.

Syntax

```
USE_SI_DENSITY = YES | NO
```

Arguments

Argument	Description
YES	Thickness variation computation is based on silicon density
NO	Thickness variation computation is based on drawn density
not present	Thickness variation computation is determined by settings of other commands

Description

This global parameter affects the operation of the THICKNESS_VS_DENSITY and POLYNOMIAL_BASED_THICKNESS_VARIATION options.

Those options also contain keywords that specify whether to use silicon or drawn dimensions. See the other reference pages for more information about the allowable combinations.

Examples

```
USE_SI_DENSITY = YES
```

See Also

- POLYNOMIAL_BASED_THICKNESS_VARIATION
- THICKNESS_VS_DENSITY

Feedback

USER_DEFINED_DIFFUSION_RESISTANCE

Specifies parameters for user-defined diffusion resistance calculation. Valid in a `CONDUCTOR` block that is defined as a gate layer.

Syntax

```
USER_DEFINED_DIFFUSION_RESISTANCE {
     GATE_TO_CONT_THRESHOLD = gc_threshold
     GATE_TO_DIFF_BEND_THRESHOLD = gd_threshold
     NUMBER_OF_TABLES = no_of_tables
     <model_1> {
          RPSQ = rpsq
          RPSQ_SHARED = rpsq_sh
          [CRT1 = crt1]
          [CTR2 = crt2]
          [T0 = T0]
          SI_GATE_WIDTH_VAR = gw
          SI_GATE_LENGTH_VAR = gl
          SI_GATE_SPACER_WIDTH = gs
          SI_CONTACT_SIZE_VAR = cs
     }
     <model_2> {
     ...
     }
}
```

Arguments

Argument	Description
gc_threshold	Distance between the gate and the contact of interest. User-defined diffusion resistance calculation is used when the layout distance is smaller than this value; standard mesh-based resistance calculation is used otherwise. Units: microns
gd_threshold	Distance between the gate and the diffusion bend. User-defined diffusion resistance calculation is used when the layout distance is smaller than this value; standard mesh-based resistance calculation is used otherwise. Units: microns
no_of_tables	Integer number of tables; mandatory if greater than 1
model_1, model_2, ...	Model name that corresponds to the `diffusion_res_model` parameter within a `conducting_layers` block in the mapping file
rpsq	Sheet resistance of diffusion between gate and shallow trench isolation Units: ohms per square

Feedback

Argument	Description
rpsq_sh	Sheet resistance of shared diffusion between two gates Units: ohms per square
crt1	Linear temperature coefficient Default: 0
crt2	Quadratic temperature coefficient Default: 0
T0	Nominal temperature Units: degrees Celsius Default: temperature specified by GLOBAL_TEMPERATURE
gw	Difference between silicon gate width and drawn gate width Units: microns
gl	Difference between silicon gate length and drawn gate length Units: microns
gs	Width of the spacer on each side of the gate Units: microns
cs	Difference between silicon contact size and drawn contact size at the bottom of the contact Units: microns

Description

For advanced processing nodes, diffusion resistance depends on factors such as contact location and diffusion layout. You can optionally take these factors into account by applying a user-defined model to specific diffusion patterns. The StarRC tool extracts layout parameters that are used as variables in a customized analysis.

Note:

This implementation of user-defined diffusion resistance is not compatible with the feature of the same name available in earlier StarRC versions. To use this feature, you must use an nxtgrd file created in StarRC version L-2016.06 or later.

To use this analysis, perform the following steps:

1. Set the USER_DEFINED_DIFFUSION_RESISTANCE command to YES in the StarRC command file. (This step is optional because the command defaults to YES.)
2. Include the USER_DEFINED_DIFFUSION_RESISTANCE command within the CONDUCTOR block of the gate conductor layer in the ITF file.

Feedback

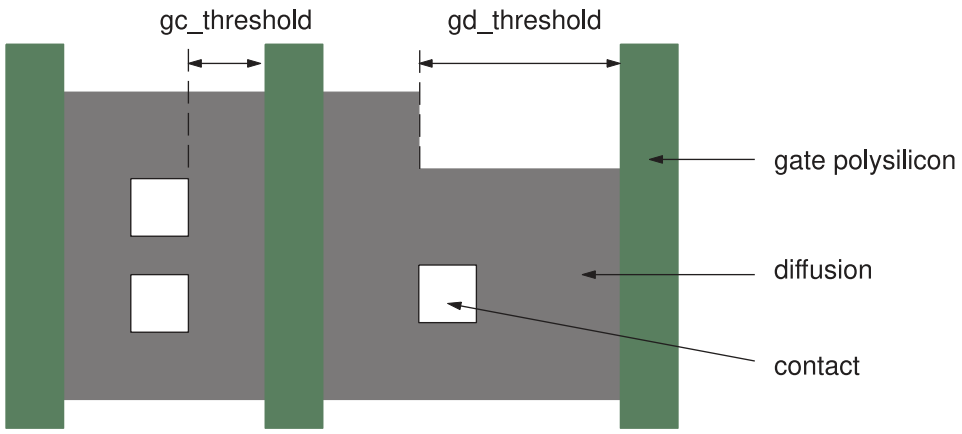
3. Set the LAYER_TYPE keyword to GATE within the CONDUCTOR block of the gate conductor layer in the ITF file.
4. Specify a model name by using the diffusion_res_model keyword in the conducting_layers command in the mapping file

Resistances calculated by the user-defined diffusion resistance method are inserted into the circuit before reduction and are included in any reduction operations. This method is used for a given contact only if the layout distances are smaller than the threshold values specified by the GATE_TO_CONT_THRESHOLD and GATE_TO_DIFF_BEND_THRESHOLD values. These parameters are illustrated in Figure 291.

You can change the user-defined diffusion resistance parameters and update the nxtgrd file with the grdgenxo -res_update command.

Simultaneous multicorner extraction is supported. However, the GATE_TO_CONT_THRESHOLD and GATE_TO_DIFF_BEND_THRESHOLD values must be identical between corners. Temperature sensitivity analysis is not supported.

Figure 291 User-Defined Diffusion Resistance Parameters



See Also

- USER_DEFINED_DIFFUSION_RES
- conducting_layers
- User-Defined Diffusion Resistance

Feedback

VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH

Models vertical gate resistance in transistor-level flows. Valid within a CONDUCTOR block.

Syntax

```
VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH {
     LENGTHS { L1  L2  … Lm }
     WIDTHS  { w1  w2  … wn }
     VALUES  { v(L1,w1) v(L2,w1) ... v(Lm,w1)
               v(L1,w2) v(L2,w2) ... v(Lm,w2)
               ...
               v(L1,wn) v(L2,wn) ... v(Lm,wn)  }
     }
```

Arguments

Argument	Description
w1, w2, ...	The width (smallest dimension) of the gate polygon, with values specified in ascending order. This dimension is the same as the gate length or channel length. Units: microns
L1, L2,…	The length (longest dimension) of the gate polygon, with values specified in ascending order. This dimension is the same as the gate width. Units: microns
v(L1,w1) ...	Vertical resistivity as a function of gate length and width Units: ohms per square The numbers in the VALUES array are interpreted on a sequential basis, independent of any carriage returns or other hidden characters.

Description

The VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH command models vertical gate resistance. This command is valid only for a conductor layer whose LAYER_TYPE option is set to GATE.

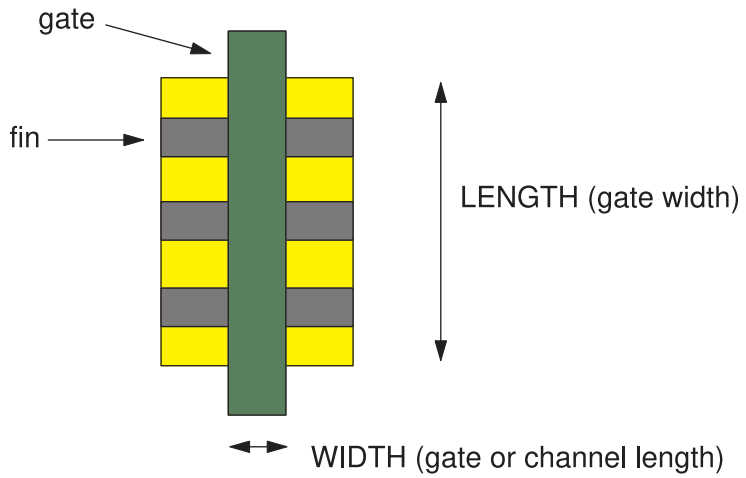
If the conductor width is larger than the maximum width in the table, the StarRC tool uses the resistivity value for the maximum width. If the conductor width is smaller than the minimum width in the table, the tool uses the resistivity value for the minimum width.

Conductor length values outside the table range are handled in a similar manner. However, the actual conductor length and width values, not the boundary values, are used to calculate the resistance.

Figure 292 shows the dimensions used for the LENGTH and WIDTH values in the table.

Feedback

Figure 292 FinFET Top View



The following usage notes apply:

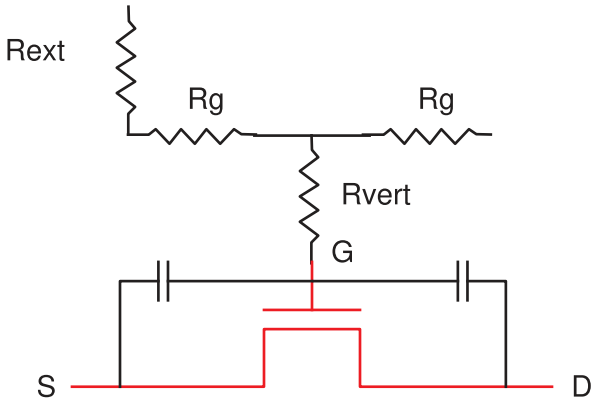
- An nxtgrd file created with the `VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH` ITF command cannot be used with the `MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE` StarRC command.
- For simultaneous multicorner (SMC) extraction, if one corner has a vertical resistance table defined with the `VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH` command, all corners must have such a table. The `LENGTH` and `WIDTH` values in all tables must be consistent between corners, but the resistivity values can be different.
- For SMC extraction with corners that have vertical resistance tables, any resistance-only etch must be the same for all corners.

In resistor tail comments, the vertical gate resistor parameter $l is the gate length, $w is the gate width, and $lvl is the gate level. The tool does not report parameters $llx, $lly, $urx, $ury, and $dir for vertical gate resistors.

The StarRC tool inserts the vertical gate resistance as shown in Figure 293. The gate node is the node below the vertical gate resistance.

If the `MOS_GATE_DELTA_RESISTANCE` command is set to `YES`, the resistor network is as shown in Figure 294.

Figure 293 Gate Resistor Node Location



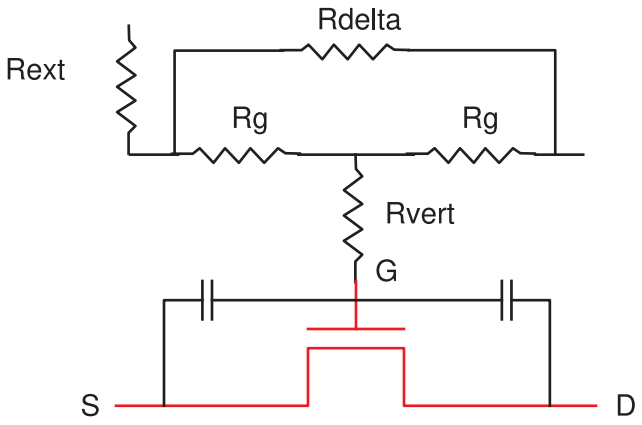
Rext: external resistance

Rg: lateral gate resistance

Rvert: vertical gate resistance

C: gate capacitance

Figure 294 Gate Resistor Node Location for Delta Resistance



Rext: external resistance

Rg: lateral gate resistance

Rvert: vertical gate resistance

C: gate capacitance

Rdelta: delta model resistance

Examples

The following example demonstrates how to use the `VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH` command to specify the vertical resistivity values shown in Table 102.

Table 105 Values for Resistance Table

Gate width (microns)	R for length=0.021 um	R for length=0.063 um	R for length=0.168 um
0.016	3010.51	3080.7	2801.4
0.020	2080.0	3000.5	2801.95

Feedback

```
CONDUCTOR GATE {
THICKNESS = 0.6 WMIN = 0.3 SMIN = 0.15
LAYER_TYPE=GATE
VERTICAL_RESISTANCE_VS_SI_WIDTH_AND_LENGTH {
   LENGTHS {0.021 0.063  0.168}
   WIDTHS {0.016 0.020}
   VALUES {3010.51  3080.7  2801.4
           2080.0   3000.5  2801.95 }
  }
}
```

See Also

- VERTICAL_GATE_RESISTANCE
- MOS_GATE_DELTA_RESISTANCE
- MOS_GATE_NON_NEGATIVE_DELTA_RESISTANCE

Feedback

VIA

Describes the properties of a via layer. The syntax depends on the type of via: standard via or trench contact virtual via.

Syntax

```
$ For all vias
VIA via_name {
     FROM = layer1
     TO = layer2
     [T0 = nominal_temp
          | T0 = nominal_temp CRT1 = lin_coeff
          | T0 = nominal_temp CRT2 = quad_coeff
          | T0 = nominal_temp CRT1 = lin_coeff CRT2 = quad_coeff
          | T0 = nominal_temp CRT_VS_AREA { … }]
     WMIN = wmin_via SMIN = smin_via
     ... }

$ Standard via
VIA via_name {
     ...
     RHO = rho_value
          | RPV = rpv_value  AREA = area_value
          | RPV_VS_AREA {…}
     [RPV_VS_COVERAGE { … } | RPV_VS_SI_COVERAGE { … } ]
     [ETCH_VS_CONTACT_AND_GATE_SPACINGS CAPACITIVE_ONLY {…}
          | ETCH_VS_WIDTH_AND_LENGTH CAPACITIVE_ONLY {…}]
     [SIDE_TANGENT = tangent_value]    }

$ Trench contact virtual via using as-drawn dimensions
VIA via_name {
     ...
     RPV_VS_WIDTH_AND_LENGTH {…}]
          | RPV_VS_AREA {…}
          | RPV = rpv_value  AREA = area_value]    }

$ Trench contact virtual via using post-etch dimensions
VIA via_name {
     ...
     ETCH_ASSOCIATED_LAYER = tc_layer
     RPV_VS_SI_WIDTH_AND_LENGTH {…}]   }

$ Trench contact virtual via with an associated silicon marker layer
VIA via_name {
     ...
     RPV_VS_WIDTH_AND_LENGTH {…}]
          | RPV_VS_AREA {…}        }

$ SYNTAX NOTE: CRT_VS_AREA cannot be used on the same layer as either
$ RPV_VS_WIDTH_AND_LENGTH or RPV_VS_SI_WIDTH_AND_LENGTH.
$ Tall contact
```

Feedback

```
VIA via_name {
      ...
      [LAYER_TYPE = TALL_CONTACT]
      [DEVICE_TYPE = device_type_name]
      [CONTACT_WIDTH_AND_LENGTH {(wmin,lmin), (w2,l2), ... (wmax,lmax)}]
      [CONTACT_TO_CONTACT_SPACING  {smin, s2 ... smax}]
      [SIDE_TANGENT = (coco_tangent, poco_tangent)]   }

$ Tall via
VIA via_name {
      ...
      [TALL_VIA_CONFIG {...} ]
```

Arguments

Argument	Description
layer1	Lower conductor layer connected by the via
layer2	Upper conductor layer connected by the via
tc_layer	Either the FROM or TO layer in the VIA definition; must be a trench contact layer defined with the LAYER_TYPE=TRENCH_CONTACT statement
lin_coeff	Layer-specific linear temperature coefficient Default: 0
quad_coeff	Layer-specific quadratic temperature coefficient Default: 0
nominal_temp	Nominal temperature for the layer Units: degrees Celsius Default: temperature specified by GLOBAL_TEMPERATURE
wmin_via	Minimum via width for merging Dimensions: microns Default: none
smin_via	Minimum via spacing for merging Dimensions: microns Default: none
rho_value	Bulk resistivity of the via or conductor layer Units: ohm-microns Default: RPV x AREA
rpv_value	Resistance per default via Units: ohms

Feedback

Argument	Description
area_value	Area of default via Units: square microns Default: 1.0e -6 / RPV
tan_value	Tangent of the contact sidewall angle. Allowed only for vias between the diffusion layer and the metal above the gate layer. Default: 0
coco_tangent	Tangent of the contact sidewall angle in the direction of contact-to-contact spacing. Allowed only for tall vias between the diffusion layer and the metal above the gate layer. Default: 0
poco_tangent	Tangent of the contact sidewall angle in the direction of poly-to-contact spacing. Allowed only for tall vias between the diffusion layer and the metal above the gate layer. Default: 0
device_type_name	Name to associate tall contact with the gate layer structure
(wmin,lmin), (w2,l2) ... (wmax,lmax)	Typical combinations of width and length values for tall contacts, in ascending order. Maximum of 9 value pairs. Units: microns
smin, s2 ... smax	Typical values of spacing between tall contacts, in ascending order. Maximum of 5 values. Units: microns

Description

The `VIA` statement describes the properties of a via layer. The syntax depends on the type of via, as follows:

- Standard via

  Most vias use this format. You must specify the two layers connected by the via and the basic resistive properties of the via. In addition, you can specify tables that modify the via resistance based on size, spacing, or coverage.

- Trench contact via

  Trench contact vias are artificial layers that enable the correct modeling of current in trench contact structures. At least one of the connected layers must be a trench contact layer. Specialized commands define the resistance of trench contact vias.

Feedback

- Tall contact

  As vias become taller, they exhibit larger coupling capacitances to other similar vias and to surrounding layers. A tall contact is usually at least 5 times as high as the gate layer thickness.

Examples

The following example is a simple via definition:

```
VIA VIA1 { FROM=M1 TO=M2 AREA=0.36 RPV=4 }
```

The following example defines several properties of a standard via:

```
VIA NMOS_CONTACT {
      FROM = NDIFF TO = M1
      AREA = 0.002
      RPV = 80.0
      CRT1 = 0.003    }
```

The following example is a trench contact via:

```
VIA VTC { FROM=POD TO=M0
     ETCH_ASSOCIATED_LAYER = M0
     RPV_VS_SI_WIDTH_AND_LENGTH {
        LENGTHS { 0.03 0.04 0.08 0.12 0.6 }
        WIDTHS { 0.02 0.03 0.04 }
        VALUES {
           360 160 80 60 22
           260 90 80 50 21
           200 80 70 40 20
        }
     }
}
```

See Also

- Tall Contact Modeling

Feedback

VIA_COVERAGE

Provides settings for via coverage analysis.

Syntax

```
VIA_COVERAGE
{
   VIA_SIZE { xmin xmax ymin ymax }
   {
      MODE {
          LANDING  (FL1 FL2 QL1 QL2 SL1 SL2)
          COVERAGE (FC1 FC2 QC1 QC2 SC1 SC2)
          [RPV (FFrpv FQrpv FSrpv FPrpv QFrpv QQrpv QSrpv QPrpv
                SFrpv SQrpv SSrpv SPrpv PFrpv PQrpv PSrpv PPrpv)]
          [VIA_MODEL_NAME (FFmodel FQmodel FSmodel FPmodel QFmodel
                QFmodel QQmodel QSmodel QPmodel SFmodel SQmodel
                SSmodel SPmodel PFmodel PQmodel PSmodel PPmodel)]
      }
      ... other modes
   }
   ... other via sizes
}
```

Arguments

Argument	Description
xmin xmax ymin ymax	Minimum and maximum x- and y-dimensions of vias for which to apply the properties enclosed within the subsequent braces Units: microns
FL1	Full coverage y-dimension for via landing Units: microns
FL2	Full coverage x-dimension for via landing Units: microns
QL1	Quarter coverage value for via landing (small enclosure value) Units: microns
QL2	Quarter coverage value for via landing (large enclosure value) Units: microns
SL1	Semi coverage value for via landing (small enclosure value) Units: microns
SL2	Semi coverage value for via landing (large enclosure value) Units: microns

Feedback

Argument	Description
FC1	Full coverage y-dimension for via coverage Units: microns
FC2	Full coverage x-dimension for via coverage Units: microns
QC1	Quarter coverage value for via coverage (small enclosure value) Units: microns
QC2	Quarter coverage value for via coverage (large enclosure value) Units: microns
SC1	Semi coverage value for via coverage (small enclosure value) Units: microns
SC2	Semi coverage value for via coverage (large enclosure value) Units: microns
FFrpv, FQrpv, ...	New RPV value for vias of the specified classification
FFmodel, FQmodel, ...	(Optional) Via variation model names of the specified classification The size of via models and RPV arrays are the same.

Description

The StarRC tool provides the capability to analyze via coverage, which is the amount by which a metal polygon that connects to a via overlaps the via polygon. Via coverage is important for reliability analysis and is analyzed separately for the top and bottom level metal polygons.

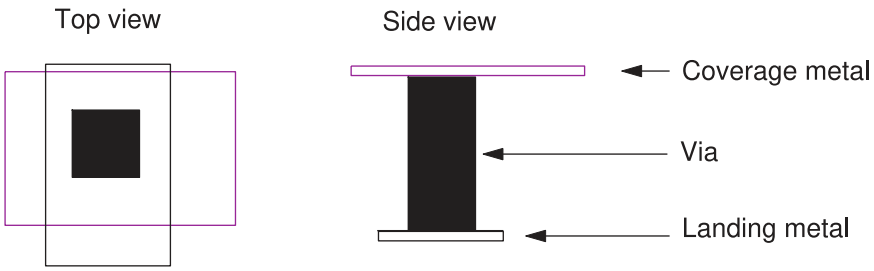
Figure 295 illustrates the relationship of the polygons of interest. The top-level metal polygon is referred to as the coverage layer or polygon and the bottom-level metal is the landing layer or polygon.

Note:

The term via coverage refers to the overall via coverage analysis technique. The terms coverage layer, coverage polygon, and coverage metal refer to the specific metal polygon in the metal layer above the via.

Feedback

Figure 295 Coverage and Landing Metal With Respect to a Via



You can enable via coverage analysis and define the coverage parameters in either of the following ways:

- Specify the `VIA_COVERAGE` command in the ITF file (nxtgrd file).
- Use one or both of the `VIA_COVERAGE` and `VIA_COVERAGE_OPTION_FILE` commands in the StarRC command file.

If the ITF file contains a `VIA_COVERAGE` command and the StarRC command file contains either the `VIA_COVERAGE` or `VIA_COVERAGE_OPTION_FILE` command, the tool issues an error message.

Examples

```
VIA VIA_1 {
   VIA_COVERAGE {
      VIA_SIZE (0.02 0.02 0.044 0.044) {
        MODE LANDING BIDIRECTIONAL {
          LANDING (0 0.014 0.012 -1 -1)
          COVERAGE (-0.001 -0.001 -0.001 -0.001 -0.001 -0.001)
          RPV (30.82 31.50 33.34 ...)
        }
      }
   }
}
```

See Also

- VIA_COVERAGE
- VIA_COVERAGE_OPTION_FILE
- Via Coverage

Feedback

WMIN

Specifies the minimum width of a geometry. Valid within a CONDUCTOR or VIA block.

Syntax

```
WMIN = width_value
```

Arguments

Argument	Description
width_value	Minimum width value Units: microns

Description

The WMIN option specifies the minimum width of a feature within a CONDUCTOR or VIA block. The WMIN value is the minimum layout dimension and should not be based on any processing steps (such as etch steps).

If you change the HALF_NODE_SCALE_FACTOR option, you must change the WMIN value accordingly. The StarRC tool does not modify the WMIN value automatically. For example, assume the WMIN value is 0.20 for a process in which the HALF_NODE_SCALE_FACTOR value is 1.0 (the default). If you set the HALF_NODE_SCALE_FACTOR value to 0.9, you should change the WMIN value to 0.18, which is calculated by applying the scale factor to the old WMIN value.

For via layers, the WMIN value is used only in transistor-level electromigration flows. The tool uses the WMIN value along with the SMIN value to calculate the maximum allowable spacing between vias for via merging. You can also provide these values by using the VIA_SMIN and VIA_WMIN commands in the StarRC command file. Values in the command file take precedence over values in the ITF file.

If the WMIN and SMIN statements are missing from the via layer definition in the ITF file and VIA_SMIN and VIA_WMIN are missing from the StarRC command file, default via merging occurs. If only one set of values is provided, the tool uses these values in the new flow. In the electromigration flow, the tool always performs via merging and ignores the MERGE_VIAS_IN_ARRAY command.

Examples

```
CONDUCTOR m1 {
     THICKNESS=1.00 WMIN=0.13 SMIN=0.15 RPSQ=0.015
}
```

Feedback

See Also

- HALF_NODE_SCALE_FACTOR
- VIA_SMIN
- VIA_WMIN
- SMIN
- Via Merging

16

Mapping Files

A design database must be accompanied by a mapping file that links each physical database layer to a process layer. This chapter describes the statements to use in a mapping file.

The following reference pages describe the mapping file statements:

- color_layers
- conducting_layers
- ignore_cap_layers
- marker_layers
- map_qtf_layers
- qtf_layers
- remove_layers
- silicon_marker_layers
- via_layers
- viewonly_layers

Comments in a Mapping File

In a mapping file, text on a line after an asterisk (*) is a comment that is not interpreted. You can begin a comment anywhere in the line. Comment lines do not wrap.

Feedback

color_layers

Maps a color layer in the layout to a modeled layer in the nxtgrd file.

Syntax

```
color_layers
  color_LyrName1 itf_LyrName1
  color_LyrName2 itf_LyrName2
...
```

Arguments

Argument	Description
color_LyrName	The color layer name in the design database
itf_LyrName	The ITF layer associated with the design color layer

Description

The `color_layers` command links a physical database color layer to a modeled layer in the nxtgrd file.

Examples

The following example maps the M1_color1 layer in the design to the metal1 ITF layer. Notice that two design layer names are mapped to metal1.

```
color_layers
M1_color1     metal1
M1_color2     metal1
M2_color1     metal2
M2_color2     metal2
```

See Also

- DPT
- DPT_COLOR_GDS_FILE
- DPT_COLOR_GDS_LAYER_MAP_FILE
- Double or Multiple Patterning Technology

Feedback

conducting_layers

Maps a conducting layer from the layout database to a conductor in the nxtgrd file.

Syntax

```
conducting_layers
db_LyrName1 itf_LyrName1 | SUBSTRATE
            | SUBSTRATE cap_scale=cscale precedence=prec_value
          [precedence=prec_value]
          [LEE_VIA_OFFSET=distance]
          [device_layer | capacitor_layer]
          [rpsq=r_value] [model=model_name]
          [mask=mask_num]
          [diffusion_res_model=diff_model_name]
          [net_segment_cut_length=cut_length]
          [overlap_fill_spacing=new_spacing]
          [table_name=name]
db_LyrName2  itf_LyrName2 ...
...
```

Arguments

Argument	Description
db_LyrNameN	Conducting layer name in the design database
itf_LyrNameN	Conducting layer name in the ITF file. Can alternatively be SUBSTRATE
cscale	Capacitance scale factor, valid only when the ITF layer is defined as SUBSTRATE Valid values: 0 or larger Default: 1 (no scaling)
prec_value	Order of precedence for different design database layers mapped to the same ITF layer. Defaults to 0 if the precedence keyword is absent.
distance	Specifies the fixed distance to via edge when the line end extension is generated if the line end is within VIA_ENCLOSURE_DISTANCE of the layer Valid range: 0.0 to any larger value Units: microns Default: -1.0 (not applied)
device_layer	Marks the layer for resistance extraction of power nets. Specify this keyword in the mapping file when you use the POWER_EXTRACT: DEVICE_LAYERS command in the command file. The device_layer keyword can be specified in any order in the mapping file for the conducting layers when the RPSQ and device model options are used.

Feedback

Argument	Description
capacitor_layer	Marks the layer as a special-purpose layer for interleaved metal-finger capacitor structures. Ground capacitances are adjusted under the assumption that polygons on this layer are part of a dense regular layout over a large area. This keyword should be used only on database layers that are used exclusively for creating interleaved capacitors. Using this keyword improves accuracy for these structures but degrades results if used on other conductor types.
r_value	Sheet resistance (in ohms per square) of the design database layer. The value specified in the mapping file overrides the RPSQ statement, RPSQ_VS_WIDTH_AND_SPACING table, or RPSQ_VS_SI_WIDTH table specified in the ITF file.
model_name	Model name to be written into the parasitic netlist, used only if the NETLIST_PARASITIC_RESISTOR_MODEL command is set to YES in the StarRC command file
mask_num	The mask ID number for multimask patterning; an integer from 1 to the value of the NUMBER_OF_MASKS keyword in the ETCH_VS_WIDTH_AND_SPACING table, inclusive
diff_model_name	Model name for user-defined diffusion resistance extraction
cut_length	Analysis length of segments on this layer Units: microns Default: 20
new_spacing	In the metal fill reuse flow, the amount by which to separate metal fill polygons from signal nets to resolve shorts Units: microns Default: none
name	Name by which to select parameter tables specified by the GATE_TO_DIFFUSION_CAP or RPSQ_VS_SI_WIDTH commands.

Description

Specify the conducting_layers section to map a conducting layer from the layout database (routing layers, device terminal layers, device layers, and so on) to a conductor layer in the nxtgrd file. You can map a database layer to the substrate by using the SUBSTRATE keyword instead of a ITF layer name.

A design database layer can appear only one time within a conducting_layers section. All keywords for that layer must be included on the same line.

Feedback

Layer Precedence

You can map multiple design database layers to a single ITF layer by using the `precedence` keyword after the ITF layer name to define the order of precedence for resolving design overlaps. However, it is preferable to avoid creating design overlaps.

Use the `precedence` keyword to specify a positive integer that establishes the layer's position in the vertical substrate profile. Larger numbers denote higher vertical precedence (a higher position in vertical space), while smaller numbers denote lower vertical precedence (a lower position). It is not necessary for precedence values to be sequential.

Substrate Layer Capacitance Scaling

You can define multiple substrate layers by mapping more than one database layer to the `SUBSTRATE` keyword. Use the `cap_scale` parameter to specify different capacitance behavior for different substrate layers. The StarRC tool first extracts the capacitance to the layer using the existing methods, then multiplies the capacitance by the specified scale factor. A scale factor of 0 results in a capacitance of 0.

The following usage notes apply:

- The `cap_scale` parameter is valid only for database layers that are specified to be substrate layers with the `SUBSTRATE` keyword.
- A substrate layer that includes the `cap_scale` parameter must also include the `precedence` parameter. However, you can use the `precedence` parameter without the `cap_scale` parameter.
- Specifying substrate layers with the same precedence value but different capacitance scaling factors is allowable. However, if these layers overlap in the design, the tool selects one of the scale factors at random. The best practice is to avoid specifying substrate layers with the same precedence value but different capacitance scaling factors.

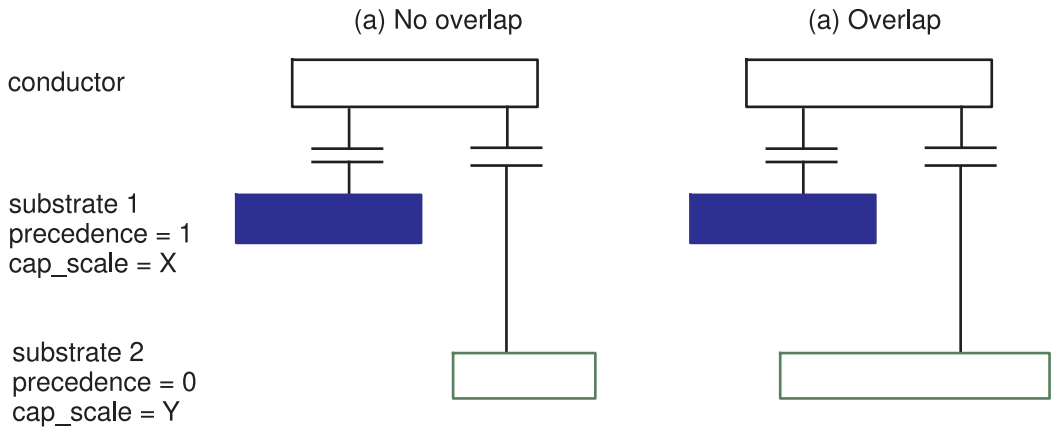
For example, Figure 296 shows a metal layer that interacts with two substrate layer. Substrate layer 1 has precedence 1, which indicates that it is vertically higher than substrate layer 2, which has precedence 0.

In part (a) of the figure, the two substrate layers do not overlap. In this case, the StarRC tool applies a scale factor of value X to all layer 1 capacitances and a scale factor of value Y to all layer 2 capacitances. If the `cap_scale` parameter is not defined, no scaling occurs for that layer.

In part (b), the two substrate layers overlap. In this case, layer 1 has higher precedence and the StarRC tool evaluates only the capacitance between the conductor layer and substrate layer 1. The interaction with substrate layer 2 is ignored.

Feedback

Figure 296 Substrate Capacitance Scaling



User-Defined Diffusion Resistance

User-defined diffusion resistance is a method that is more accurate for advanced process nodes than the standard mesh resistance calculation.

The `diffusion_res_model` keyword has an effect only under the following conditions:

- The `USER_DEFINED_DIFFUSION_RES` command in the StarRC command file is set to `YES` (or is omitted, because it defaults to `YES`).
- The `CONDUCTOR` block in the ITF file for the gate conductor layer contains a `USER_DEFINED_DIFFUSION_RESISTANCE` command. The gate conductor layer must have its `LAYER_TYPE` keyword set to `GATE`.

If the diffusion resistance calculation is entirely embedded in the SPICE model for certain device types, do not use the `diffusion_res_model` keyword in the mapping file.

Examples

A simple `conducting_layers` block is as follows:

```
conducting_layers
  m2      metal2
  m1      metal1
  nsd     diffusion
  psd     diffusion
```

The following example assigns some properties in addition to mapping the layers:

```
conducting_layers
  M2            metal2
  M1            metal1
  POLY          fpoly
  nsd           diffusion     device_layer  RPSQ=0.5
  psd           diffusion     device_layer
```

```
  welltie       SUBSTRATE
  ngate         gpoly         device_layer
  pgate         gpoly
```

The following example shows that the metal 1 layer is split into two masks named M1_color1 and M1_color2. The metal 2 layer is not split; do not use the `MASK` keyword in this case.

```
conducting_layers
  M1_color1      M1 MASK=1
  M1_color2      M1 MASK=2
  M2             M2
```

The following example shows a mapping file that specifies different parameter tables for different layers. When the `table_name` parameter is defined for a layer, the StarRC tool uses the tables with same name specified in the `GATE_TO_DIFFUSION_CAP` command (for gate-to-diffusion capacitance) and the `RPSQ_VS_SI_WIDTH` command (for sheet resistance).

```
conducting_layers
  ngate   gpoly  table_name=NMOS
  pgate   gpoly  table_name=PMOS
  tngate  gpoly
  tpgate  gpoly
```

The following example shows a mapping file that specifies different user-defined diffusion resistance model names for different layers. To calculate resistance, the StarRC tool uses the parameters defined for that model name in the `USER_DEFINED_DIFFUSION_RESISTANCE` command in the ITF file for the gate conductor.

```
conducting_layers
  gate_n   GATE  diffusion_res_model=res_n
  gate_p   GATE  diffusion_res_model=res_p
```

The following example shows a mapping file that specifies that the extended line end should be 0.03 microns to via edge if the original line end is within `VIA_ENCLOSURE_DISTANCE` of layer M1:

```
conducting_layers
   M1 metal1 LEE_VIA_OFFSET=0.03
```

See Also

- POWER_EXTRACT : DEVICE_LAYERS
- ETCH_VS_WIDTH_AND_SPACING
- GATE_TO_DIFFUSION_CAP
- RPSQ_VS_SI_WIDTH

Feedback

- USER_DEFINED_DIFFUSION_RESISTANCE
- USER_DEFINED_DIFFUSION_RES
- User-Defined Diffusion Resistance
- The Metal Fill Reuse Flow
- LINE_END_EXTENSION_TABLE

Feedback

ignore_cap_layers

Specifies to ignore the capacitance between specified design database layers.

Syntax

```
ignore_cap_layers
   db_LyrName1 db_LyrName2 [db_LyrName3 …] [L=length]
   db_LyrName4 db_LyrName5 [db_LyrName6 …] [L=length]]
...
```

Arguments

Argument	Description
db_LyrName1, db_LyrName2, ...	Design database layer names
length	A positive value specifying the distance over which the interlayer capacitance should be ignored Units: microns
NGATE NSD L=*value*	The n-diffusion length for which you want to ignore the capacitance to the particular NGATE source or drain. Valid only for NMOS transistors. If a layer pair is not part of a MOS definition, a warning is issued.
PGATE PSD L=*value*	The p-diffusion length for which to ignore the capacitance to the particular NGATE source or drain. Valid only for PMOS transistors. If a layer pair is not part of a MOS definition, a warning is issued.
nsd SUBSTRATE L=*value*	The n-diffusion to substrate length for which to ignore the capacitance
psd SUBSTRATE L=*value*	The p-diffusion to substrate length for which to ignore the capacitance

Description

The `ignore_cap_layers` section specifies that the capacitance between certain design database layers is to be ignored. In addition, you can specify a length value (or distance) of diffusion where capacitances are ignored.

To partially ignore capacitances for source and drain transistor areas to gate and substrate, add the optional `L` keyword with a length value. In this case, the StarRC tool ignores the capacitance up to the specified amount. The total capacitance on the net with specified capacitance increases when an L value is specified.

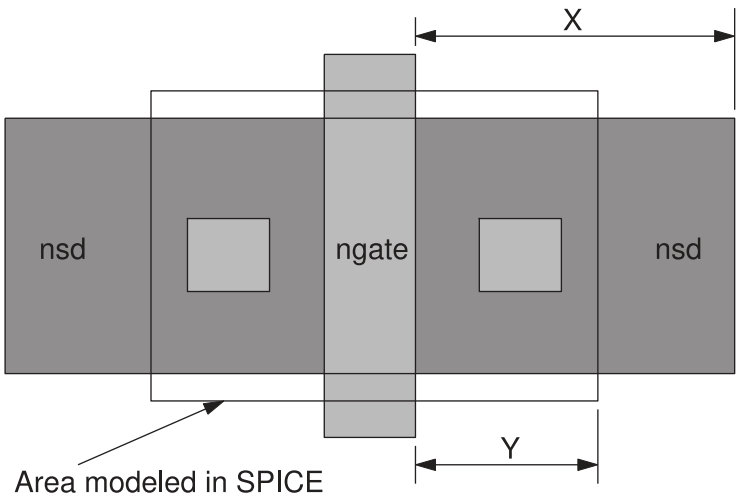
Feedback

- All parallel-plate and fringing capacitance components between a specified layer pair are omitted from the parasitic netlist.
- If you specify a design database layer in the `ignore_cap_layers` section of the mapping file, this function acts independently from the `IGNORE_CAPACITANCE` command in the StarRC command file, regardless of the `IGNORE_CAPACITANCE` command setting.
- If more than two layers are specified, all the capacitances between every possible combination of layers are ignored. To ignore the capacitance between polygons of the same layer, specify the same layer twice.

The field solver `FSCOMPARE` and `FS_EXTRACT_NETS` modes interpret a specified `ignore_cap_layers` section when producing field solver results for accuracy validation or netlist creation.

The StarRC tool ignores all capacitance between a gate to a source or drain terminal of a metal oxide semiconductor (MOS) transistor. This is acceptable when the complete capacitance of that MOS transistor is present in the SPICE model. However, in some cases the drawn devices have larger source or drain areas than the characterized transistors, as shown in Figure 297. Ignoring them completely during extraction is inaccurate.

Figure 297 Applying Capacitance to MOS Source and Drain



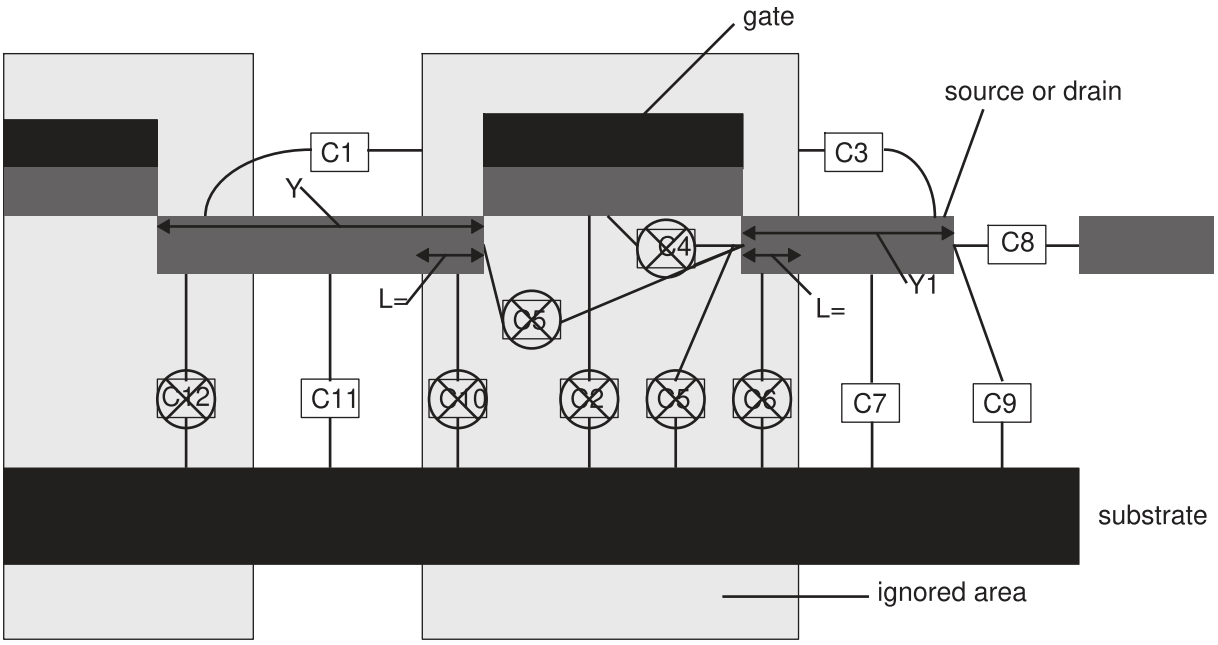
To avoid double counting the capacitance already modeled in the device, you can specify partial ignoring of gate capacitance. See Figure 298. To do this, set a length parameter in the ignore cap section of your mapping file for the specified layer pair using the `ignore_cap_layers` section.

Feedback

The conditions under which the length parameter is supported are as follows:

- Both layer1 and layer2 of the specified layer pair should be part of the MOSFET device. One layer might be the gate terminal layer or substrate while the other is the source or drain terminal layer.
- The length must be positive.

Figure 298 Defining a Partial Distance to Ignore



Examples

The following shows a simple example:

```
ignore_cap_layers
   tn_diff       NSD   nnsd
   m1_cap_term   SUBSTRATE
```

The following example specifies a length value for source and drain of the MOS transistor to partially ignore capacitance, which can cause the total capacitance on a net to increase.

```
ignore_cap_layers
  ngate NSD L=0.1
  pgate PSD L=0.1
  nsd SUBSTRATE L=0.1
  psd SUBSTRATE L=0.1
```

Feedback

marker_layers

Specifies marker layers.

Syntax

```
marker_layers
   db_LyrName1
   db_LyrName2
...
```

Arguments

Argument	Description
db_LyrName	Name of marker layer in the design database

Description

Marker layers are optional objects derived from text interactions to identify special nets that become either primary input or output ports or SKIP_CELLS ports (instance pins) in the parasitic netlist.

Note:

Marker layers are not valid for the Calibre Connectivity Interface flow.

Examples

```
marker_layers
   metal1_pin
   poly_pin
   metal7_pio
```

See Also

- SKIP_CELLS
- MARKER_GENERATION

Feedback

map_qtf_layers

Specifies mapping layers for QTF files. Used only in a QTF flow.

Syntax

```
map_qtf_layers
   db_LyrName    qtf_LyrName
   db_LyrName    qtf_LyrName
...
```

Arguments

Argument	Description
db_LyrName	Name of layer in the design database
qtf_LyrName	Name of layer in the QTF file

Description

The `map_qtf_layers` and `qtf_layers` sections of the mapping file are used only for the QTF flow; both sections are mandatory for this flow.

The `map_qtf_layers` section contains statements that map input database layers to QTF layer aliases. A QTF layer alias is defined in the QTF file by the `layerAlias` command in the gds2cap utility, which is provided with the QuickCap tool package.

The `qtf_layers` section is a list of QTF layers to be considered during extraction. If a database layer mapping is included in the `map_qtf_layers` section but the QTF layer name does not appear in the `qtf_layers` section, the database layer mapping to this QTF layer is ignored.

The `map_qtf_layers` section must appear before the `qtf_layers` section. In addition, database layers specified in the `map_qtf_layers` section of a mapping file must not be listed in the `remove_layers` section. If a database layer is listed in both the `map_qtf_layers` and `conducting_layers` sections of the mapping file, the definition in the `map_qtf_layers` section takes precedence.

Examples

```
map_qtf_layers
   fin_lyr     fin_lyr
   npoly1      n_finpoly1
   ppoly1      p_finpoly1
...
qtf_layers
   fin_lyr
   n_finpoly1
```

```
   p_finpoly1
...
```

See Also

- FS_QTF_FILE
- qtf_layers
- The QTF Flow

Feedback

qtf_layers

Specifies layers to use for QTF extraction. Valid only in a QTF flow.

Syntax

```
qtf_layers
   qtf_LyrName1
   qtf_LyrName2
...
```

Arguments

Argument	Description
qtf_LyrName1, qtf_LyrName2, ...	Name of QTF layers

Description

The `map_qtf_layers` and `qtf_layers` sections of the mapping file are used only for the QTF flow; both sections are mandatory for this flow.

The `map_qtf_layers` section contains statements that map input database layers to QTF layer aliases. A QTF layer alias is defined in the QTF file by the `layerAlias` command in the gds2cap utility, which is provided with the QuickCap tool package.

The `qtf_layers` section is a list of QTF layers to be considered during extraction. If a database layer mapping is included in the `map_qtf_layers` section but the QTF layer name does not appear in the `qtf_layers` section, the database layer mapping to this QTF layer is ignored.

The `map_qtf_layers` section must appear before the `qtf_layers` section. In addition, database layers specified in the `map_qtf_layers` section of a mapping file must not be listed in the `remove_layers` section. If a database layer is listed in both the `map_qtf_layers` and `conducting_layers` sections of the mapping file, the definition in the `map_qtf_layers` section takes precedence.

Examples

```
map_qtf_layers
   fin_lyr     fin_lyr
   npoly1      n_finpoly1
   ppoly1      p_finpoly1
...
qtf_layers
   fin_lyr
   n_finpoly1
```

```
   p_finpoly1
...
```

See Also

- FS_QTF_FILE
- map_qtf_layers
- The QTF Flow

Feedback

remove_layers

Specifies layers to be ignored by the extraction tool.

Syntax

```
remove_layers
   db_LyrName1
   db_LyrName2
...
```

Arguments

Argument	Description
db_LyrName	Name of the design database layer

Description

The `remove_layers` section specifies layers to be ignored during extraction.

You can use wildcards in the `remove_layers` section of the mapping file, as follows:

- Use the question mark (?) to represent a single character.
- Use the pound sign (#) to represent a string of any length.

These wildcards do not apply to other sections of the mapping file. Note that an asterisk (*) begins a comment in a mapping file command and therefore cannot be used as a wildcard character.

Note:

Removing database layers can affect the connectivity of the output parasitic netlist. A typical extraction does not require the use of this mapping file section.

Examples

```
remove_layers
   ntap
   ptap
   tndiff
   tpdiff
```

See Also

- MAPPING_FILE

Feedback

silicon_marker_layers

Specifies silicon marker layers.

Syntax

```
silicon_marker_layers
   db_LyrName1  [target_layer] [ITF = new_layer [MASK = maskID]]
   db_LyrName2 ...
...
```

Arguments

Argument	Description
db_LyrName1	Marker layer in the design database
target_layer	Target layer in the design database; must be either a trench contact virtual via layer or a conductor layer
new_layer	Database layer to use instead of the target layer when the target layer is overlapped by the marker layer
maskID	Mask number. If not specified, the mask ID is inherited from the target layer.

Description

Silicon marker layers have no connectivity. Instead, they provide information to be used in conjunction with another database layer to affect the processing of that database layer. In other words, the properties of a database layer can be different depending on whether the layer overlaps with a silicon marker layer.

The following applications are supported:

- Trench Contacts
- Covertical Conductor Layers
- Layout-Only Devices

Trench Contacts

You can use silicon marker layers to represent the silicon dimensions of trench contact virtual vias. For this usage, each line in the `silicon_marker_layers` section must contain only two fields: the name of the marker layer followed by the name of the trench contact layer. Do not use the third or fourth fields in the mapping file (the items indicated by the ITF and MASK keywords).

A silicon marker layer provides the area of a trench contact virtual via for the purpose of calculating accurate contact resistance. The marker layer has no effect on capacitance extraction. The shapes of the trench contacts and the silicon marker layers must be simple rectangles.

In the ITF file, the associated trench contact virtual via layer must contain one of the RPV_VS_AREA or RPV_VS_WIDTH_AND_LENGTH tables.

If the StarRC command file sets the NETLIST_TAIL_COMMENTS command to YES, the reported area of a via is the original trench contact fake via area, not the marker shape area.

The following is an example of a silicon marker layer specification for a trench contact.

```
silicon_marker_layers
   TAP1_marker   TAP1
   TAP2_marker   TAP2
```

Covertical Conductor Layers

You can use a silicon marker layer to change the mapping of one database conductor layer to another database conductor layer if the two layers are associated by using the ASSOCIATED_CONDUCTOR keyword in their respective CONDUCTOR blocks. For example, the following line remaps the portion of layer M0_A that overlaps marker layer M0_logic1 to layer M0_thin:

```
silicon_marker_layers
   M0_logic1    M0_A   ITF = M0_thin
```

You can also use a silicon marker layer to specify a mask number that affects the layer properties through other ITF commands. The following line remaps the portion of layer M0_A that overlaps marker layer M0_memory to layer M0_thick. In addition, a mask ID of 5 is used in any ITF command that specifies a mask-dependent thickness variation command.

```
silicon_marker_layers
   M0_memory    M0_A   ITF = M0_thick   MASK = 5
```

The following usage notes apply:

- The layout versus schematic (LVS) tool database layers must be defined in the conducting_layers section of the mapping file.
- If one portion of the LVS database layer is covered by multiple marker layers in the layout, the StarRC tool issues an error message.
- If the ITF= keyword is not specified, the ITF layer defaults to the ITF layer specified in the conducting_layers statement for the same database layer.
- If the MASK= keyword is not specified, the mask ID is inherited from the target layer.

- If the MASK= keyword is specified in the mapping file and the conductor layer uses the ETCH_VS_WIDTH_AND_SPACING command, but the mask ID is not defined with the MASK keyword of the ETCH_VS_WIDTH_AND_SPACING command, the StarRC tool issues an error message.
- If the MASK= keyword is specified in the mapping file and the conductor layer uses the THICKESS_VARIATION_VS_MASK command, but the mask ID is not present in the THICKESS_VARIATION_VS_MASK command, the nominal thickness is used.
- For simultaneous multicorner extraction, if two routing layers are associated with each other in one corner, they must be associated with each other in all corners. The types of thickness variation on associated routing layers must be identical for all corners in terms of variation commands and number of masks; however, the thickness variation values can be different.
- The bottom heights of all M0 conductors must be identical. The dielectric layer profile under all M0 layers must be identical.
- If a mask ID is specified, the output netlist shows the mask level for extracted resistors that are covered by the marker layer.

Layout-Only Devices

Layout versus schematic (LVS) tools sometimes generate new layer names to represent the layers of layout-only devices. However, the StarRC tool needs to associate these new layer types with layers that are defined in the ITF file.

You can use a silicon marker layer to associate a layer name generated by an LVS tool with an ITF layer. In the following example, LVS layers n_layout and p_layout are assigned to ITF layers n_x1 and p_x2:

```
silicon_marker_layers
   n_layout    ITF = n_x1
   p_layout    ITF = p_x2
```

See Also

- RPV_VS_AREA
- RPV_VS_WIDTH_AND_LENGTH
- NETLIST_TAIL_COMMENTS
- ETCH_VS_WIDTH_AND_SPACING
- THICKNESS_VARIATION_VS_MASK

Feedback

via_layers

Maps a via layer in the database to a via layer in the nxtgrd file.

Syntax

```
via_layers
    db_LyrName1 itf_LyrName1 [device_layer]
         [RPV = rpv_value] [AREA = via_area]
         [MODEL=model_name]
         [MAX_VIA_ARRAY_LENGTH = length]
         [MAX_VIA_ARRAY_SPACING = spacing]
         [REDUCE_VIA = reduction_option]
    db_LyrName2 itf_LyrName2 ...
...
```

Arguments

Argument	Description
db_LyrName	Via layer in the design database
*itf_LyrNam*e	Via layer in the in the technology file, ITF, or nxtgrd file
device_layer	Layer for the resistance extraction of power nets. Specify this keyword when you use the POWER_EXTRACT:DEVICE_LAYERS command in the command file. The device_layer keyword can be specified in any order in the mapping file for the via layers when RPV, device model, or via merging options are used.
rpv_value	Resistance per via Units: ohms per via
via_area	Area of the via Units: square microns
model_name	String representing a model name, used to write out model names for parasitic resistors in the parasitic netlist. All via layers must be mapped to a model if you specify the NETLIST_PARASITIC_RESISTOR_MODEL command. If you have not specified a corresponding resistor MODEL in the database, no model is printed to the parasitic netlist for that resistor and the StarRC tool issues a warning in the summary file.
length	The maximum length of a via array to be reduced to a single via. If you do not specify this argument, an array with via spacing less than or equal to *spacing* is reduced to one via and eventually to one resistor. Typically, set the value of *length* to the DRC spacing rule for that via layer. Units: microns Default: 20

Feedback

Argument	Description
spacing	The maximum allowable spacing between vias for via merging. Units: microns Default: none
reduction_option	The reduction specification; used only if the KEEP_VIA_NODES command is set to MAPPING_FILE Valid values: yes (the default), no, power, signal

Description

The via_layers section maps a via layer in the database to a via layer in the nxtgrd file.

You can override the via resistance value contained in the nxtgrd file to which the database is being mapped by specifying the RPV value and the via area. The layers appearing in this category can only be nxtgrd file via layers.

You can specify how reduction is applied to vias by setting the KEEP_VIA_NODES command to MAPPING_FILE in the command file and including the REDUCE_VIA option in the via_layers section in the mapping file. The reduction options are as follows:

- yes (reduce vias)
- no (do not reduce vias)
- power (reduce vias only if they belong to power nets)
- signal (reduce vias only if they belong to signal nets)

You can use the MAX_VIA_ARRAY_LENGTH and MAX_VIA_ARRAY_SPACING options to customize via merging behavior. Via merging is also affected by the MERGE_VIAS_IN_ARRAY command in the StarRC command file.

Examples

This example provides new RPV and area values for two of the four via layers:

```
via_layers
    V2       via2
    V1       via1
    SubCont  Cont rpv=10 area=0.04
    PolyCont Cont rpv=8  area=0.04
```

This example marks two via layers as device layers:

```
via_layers
    V1         via1
    POLY_CONT polyCont device_layer
    DIFF_CONT diffCont device_layer rpv=0.5
```

Feedback

See Also

- POWER_EXTRACT : DEVICE_LAYERS
- RPV_VS_AREA
- MERGE_VIAS_IN_ARRAY
- Via Merging

Feedback

viewonly_layers

Specifies view-only layers to be written to the StarRC extracted view.

Syntax

```
viewonly_layers
   db_LyrName1
   db_LyrName2
...
```

Arguments

Argument	Description
db_LyrName	Name of the design database layer

Description

The `viewonly_layers` section specifies the view-only (nonconnectivity) layers to be written to the StarRC extracted view. The layers specified in this section are written to the extracted view in the same way as the connectivity layers.

To make the view-only layers visible in the OpenAccess (OA) view, you must also map these layers in the layer purpose pair (LPP) mapping file.

Examples

The following example shows a portion of a StarRC layer mapping file:

```
remove_layers
   nwdiode25_dio
   nwdiode33_dio
viewonly_layers
   medvtp
```

The following example shows the corresponding LPP mapping file:

```
M5PIN poly drawing poly net poly subnode
M6PIN poly drawing poly net poly subnode
M7PIN poly drawing poly net poly subnode
M8PIN poly drawing poly net poly subnode
M9PIN poly drawing poly net poly subnode
POLYPIN poly drawing poly net poly subnode
medvtp medvtp drawing nil nil nil nil
```