# StarRC™ Parasitic Explorer User Guide

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 Parasitic Explorer user guide describes how to use the StarRC Parasitic Explorer tool.

This manual describes how to use the StarRC Parasitic Explorer tool to understand and report parasitics that have been extracted by the StarRC tool and stored in a GPD Parasitic Database.

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 Parasitic Explorer 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:

- StarRC™ User Guide and Command Reference
- PrimeTime® Suite
- Custom Compiler™
- Using Tcl With Synopsys Tools

Feedback

## 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.<br>• 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. |

## 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.

# 1

# Overview

The Parasitic Explorer tool helps you query parasitic resistors and capacitors stored in a parasitic database (GPD) created by the StarRC extraction tool.

The overview of the Parasitic Explorer tool includes the following topics:

Feedback

# Parasitic Explorer Features

The StarRC Parasitic Explorer tool provides methods for exploring the contents of a GPD, which is a compact and efficient binary database that contains design parasitics extracted by the StarRC tool. In the GPD, parasitic resistors and capacitors are considered to be design objects that can be handled in ways similar to other design objects.

The Parasitic Explorer tool is a Tcl (tool command language) environment with a prompt of starrc_shell. You can execute Tcl commands in the shell interactively. Alternatively, you can write scripts to automate tasks.

For both gate-level and transistor-level parasitic explorer flows, you can use the Tcl shell or the GUI to perform tasks such as the following:

- In the Tcl shell:
  - Create a collection of parasitic resistors, ground capacitors, or coupling capacitors from one or more nets
  - Query parasitic element attributes such as resistance, capacitance, subnode name, layer name, layer number, and physical location
  - Report properties of the data in the GPD such as completeness, the StarRC version used to perform the extraction, the presence or absence of specific types of data, and the number of nets, cells, and ports
  - Report the corner names and layer names defined in the GPD
- In the GUI:
  - Annotate parasitics on specific nets for easy visualization
  - Visualize opens and shorts for debugging

Usage requirements are as follows:

- The GPD must be created by StarRC version O-2018.06-SP4-1 or later.
- You must have the Parasitic Explorer license and either the StarRC Ultra or StarRC Ultra+ license. You cannot use combinations of other StarRC licenses.
- You must have the Custom Infrastructure license that is part of the Custom Compiler product family to use the `starrc_explorer` command (the standalone Custom Compiler interface).

# Parasitic Explorer Documentation

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

- Command information displayed with the `help` command
- Man pages displayed with the `man` command
- Help in the graphical user interface
- The *StarRC User Guide and Command Reference* user guide, available on SolvNetPlus
- The *Using Tcl With Synopsys Tools* user guide, available on SolvNetPlus

## Command Help

The `help` command provides concise information about Parasitic Explorer commands. You can display a list of commands or view the syntax of a specific command.

The `help` * command shows a list of commands, organized by command group:

```
starrc_shell> help *
 ...
Default Command Group:
 add_to_collection, all_inputs, all_instances
 ...
```

You can use wildcards to restrict the scope of the list or to find the name of a command that you cannot remember exactly. For example, to find all commands that contain the string “capacitor,” enter

```
starrc_shell> help *capacitor*
 get_ground_capacitors            # Find parasitic ground capacitors
 get_coupling_capacitors          # Find parasitic coupling capacitors
 ...
```

For a concise description of a command, enter `help` with the command name:

```
starrc_shell> help get_ground_capacitors
 get_ground_capacitors   # Get ground capacitor collection objects
```

To see the full command syntax, including options and arguments, use the `-verbose` option:

```
starrc_shell> help get_ground_capacitors -verbose
 get_ground_capacitors   # Get ground capacitor collection objects
   [-filter expression]     (Filter collection with 'expression')
   [-quiet]                 (Suppress all messages)
   [-parasitic_corners corner_name]  (Parasitic corner selection)
```

Feedback

```
   [-all_parasitic corners] (Select all parasitic corners)
   [-of_objects objects]    (Get ground capacitors of these nets)
   [-from_node from_node]   (From pin, port, or net internal node)
   [-to_node to_node]       (To pin, port, or net internal node)
```

An alternate method to display the same information is to enter the command name directly with the `-help` option:

```
starrc_shell> get_ground_capacitors -help
 get_ground_capacitors   # Get ground capacitor collection objects
   [-filter expression]     (Filter collection with 'expression')
   [-quiet]                 (Suppress all messages)
   [-parasitic_corners corner_name]  (Parasitic corner selection)
   [-all_parasitic corners] (Select all parasitic corners)
   [-of_objects objects]    (Get ground capacitors of these nets)
   [-from_node from_node]   (From pin, port, or net internal node)
   [-to_node to_node]       (To pin, port, or net internal node)
```

## Man Pages

To find descriptive information about a command, variable, or system message, use the `man` command at the `starrc_shell>` prompt during a Parasitic Explorer session. Type `man` followed by the command name, variable name, or message code.

Man pages for commands follow a standard format that includes the syntax, a description of each option and argument, a general description of the command and its usage, examples, and a list of related commands and variables.

Man pages for variables show the name, value type (string, list, Boolean, integer, or floating-point number), the default, and a description of the variable and its effects.

Man pages for error, warning, and information messages include the name, a brief description, and some suggestions for followup actions. To view the man page for an error message, use the `man` command with the message code. Type uppercase letters for the error code.

**Note:**

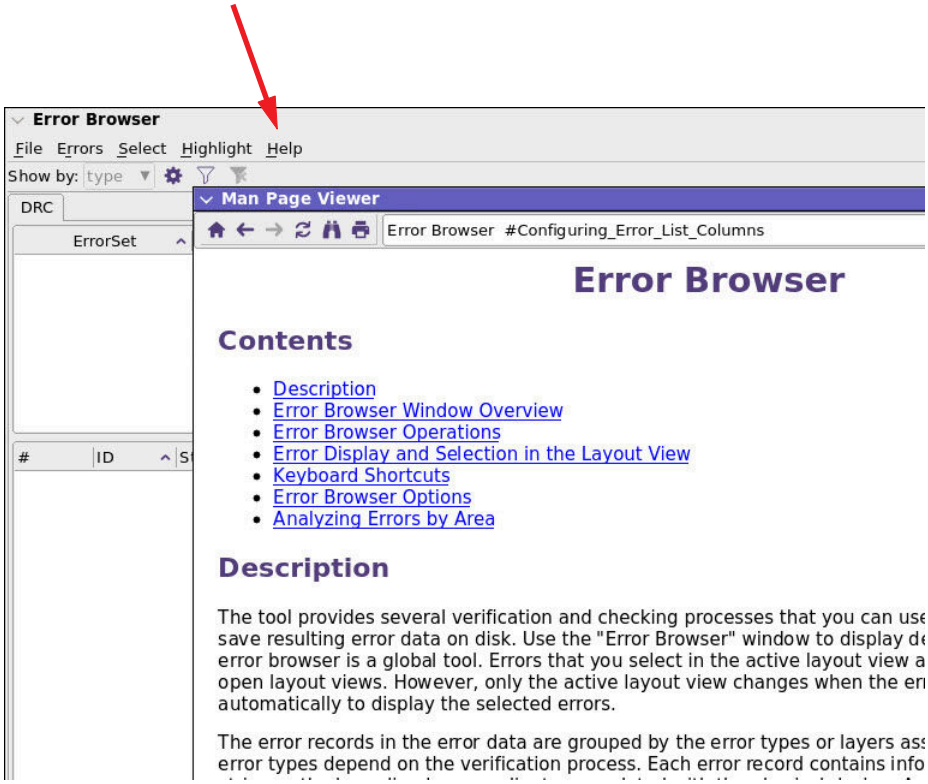
Some man pages are shared with the PrimeTime static timing analysis tool. Some information in the man pages might not be valid for the Parasitic Explorer tool.

Feedback

## User Interface Help

When you are using the Parasitic Explorer GUI, you can find general information about the UI layout and features by clicking **Help**, as shown in Figure 1.

*Figure 1 Example of User Interface Help*



Feedback

# Parasitic Explorer Session Management

The Parasitic Explorer tool runs under the Linux operating system. Before you can use it, the application must be installed and licensed at your site.

To start an interactive session, enter the `starrc_shell` command at the operating system prompt.

The Parasitic Explorer tool checks out a StarRC license and displays an initial message and the `starrc_shell` prompt. Here is an example, but the message you see might be different depending on the version.

```
                              StarRC
    Version V-2023.12 for linux64 - December 9, 2019
              Copyright (c) V-2023.12 by Synopsys, Inc.

This software and the associated documentation are proprietary to
to Synopsys, Inc. This software may only be used in accordance with
the terms and conditions of a written license agreement with Synopsys,
Inc. All other use, reproduction, or distribution of this software is
strictly prohibited.

starrc_shell>
```

To end a Parasitic Explorer session, enter the `quit` or `exit` command at the prompt:

```
starrc_shell> exit
Maximum memory usage for this session: 0.72 MB
CPU usage for this session: 0 seconds
Diagnostics summary: 2 errors

Thank you for using starrc_shell!
%
```

## The Command Log File

The Parasitic Explorer tool saves the session history in the command log file. This file contains all of the commands executed during the session and serves as a record of your work. You can repeat the session by running the file as a script, using the `source` command.

The log file is named starrc_shell_command.log and is located in the current working directory. A new log file overwrites an existing log file with the same name. Before you start a new session, rename any log files that you want to keep.

You can specify a name for the command log file by setting the `sh_command_log_file` variable in a setup file. You cannot change this variable during a session.

Feedback

# The Parasitic Explorer Shell Interface

The `starrc_shell` interface is based on the Tcl scripting language. You can use features of Tcl such as user-defined variables, procedures, conditional execution, lists, and expressions.

The command syntax is case-sensitive. Commands, command options, arguments, and variables generally consist of lowercase characters.

Object names in the design are also case-sensitive. For example, the names clk and CLK refer to two different design objects.

A detailed description of the features of Tcl is beyond the scope of this user guide. For more information, see *Using Tcl With Synopsys Tools*, which is available on SolvNetPlus, or a reference book on Tcl.

The prompts are programmable. By default, the primary prompt is `starrc_shell>` and the secondary prompt is a question mark ( `?` ). To change the prompt, set the `tcl_prompt1` or `tcl_prompt2` variable to the name of a procedure that displays the new prompt. The procedure cannot take an argument. For example, to make the primary prompt an asterisk ( `*>` ), do the following:

```
starrc_shell> proc prompt1 {} { echo -n "*> " }
starrc_shell> set tcl_prompt1 prompt1
prompt1
*>
```

## Entering Commands Interactively

You can abbreviate command names and options to the shortest unambiguous string. For example, you can abbreviate the `get_attribute` command to `get_attr`.

Using command abbreviations is convenient for interactive sessions. However, avoid using abbreviations in scripts, because command changes in later releases might make the abbreviations ambiguous.

The `sh_command_abbrev_mode` variable determines whether command abbreviation is enabled. The default is `Anywhere`; you can also set the variable to `Command-Line-Only`. To disallow all command abbreviation, set the `sh_command_abbrev_mode` variable to `None`.

If you enter an ambiguous command, the tool attempts to help you find the correct command. For example, the `all_in` command as entered here is ambiguous:

```
starrc_shell> all_in
Error: ambiguous command 'all_in' matched 2 commands:
       (all_inputs, all_instances) (CMD-006)
```

Feedback

The error message lists up to three possible matches. To list all of the commands that match the ambiguous abbreviation, use the help function with a wildcard pattern. For example,

```
starrc_shell> help all_in_*
 all_inputs        # Create a collection of all input ports in a design
 all_instances     # Create a collection of all instances of a design
```

You can split long commands across multiple lines by using the backslash (\) continuation character or by clicking the Enter key while a command is still incomplete. In this case, the tool displays the secondary prompt for each additional line of the command. The default secondary prompt is a question mark. For example,

```
starrc_shell> alias my_cap_report {get_ground_capacitors \
? -of_objects list_of_nets}
```

In this user guide, a command that cannot fit on one line is shown on multiple lines with the continuation character. However, the secondary prompt is omitted from the examples.

## Using Command Scripts

A command script is a text file containing a sequence of commands. Create scripts to carry out complex or repetitive tasks. The log file generated at the end of an interactive session can also be used as a script.

The Parasitic Explorer tool recognizes script files in plain ASCII format, ASCII compressed in gzip format, and ASCII encoded into bytecode format by the TclPro Compiler. To execute a script in any of these forms, use the `source` command:

```
starrc_shell> source file_name
```

To execute a script upon startup, use the `-file` option (short form `-f`):

```
% starrc_shell -f file_name
```

You can create scripts that use variables, loops, and conditional execution. The flow control commands `if`, `while`, `for`, `foreach`, `break`, `continue`, and `switch` determine the execution order of other commands.

Any line of text in a script file that begins with the pound sign (`#`) is a comment. Any text from a semicolon and pound sign (`;#`) to the end of a line is also considered to be a comment.

You can redirect the output to a file. The following command runs the Tcl script named rc_analysis.tcl and redirects all output and error messages to the file result_file.out.

```
% starrc_shell -file rc_analysis.tcl > result_file.out
```

Feedback

If your script contains a syntax error, the tool stops and waits for input unless the `sh_continue_on_error` variable is set to `true`.

End the script with the `quit` or `exit` command. Otherwise, the `starrc_shell` prompt does not appear, and you do not know when the script has finished executing. If your script does not end with the `quit` command, the tool waits for input. Type `quit` or `exit` to end the session.

## Tcl Commands

Commands are statements that cause actions, such as defining values, executing analysis, or displaying reports. The result of the command is displayed. When there is no specific resulting output, commands return a 1 to indicate success and a 0 to indicate failure. For example:

```
starrc_shell> read_parasitics -keep_capacitive_coupling -format gpd gpd
1
```

Command examples in this user guide do not always show the return value.

For some commands, the result is a collection. For example, the result of the `get_ports` command is a collection of ports. The following command creates a collection of all ports whose names begin with the letters IN .

```
starrc_shell> get_ports IN*
{"IN1", "IN2", "IN3", "IN4"}
```

After the command executes, the collection handle is displayed. The collection handle is an automatically-generated name for the collection of objects created by the command. If you want to use the objects in additional operations, set the collection to a variable or nest it within another command.

Enclose each nested command in square brackets. For example, the `report_attribute` command lists the attributes attached to one or more specified input ports. The following example creates a collection of input ports with the `get_ports` command and passes the result to the `report_attribute` command:

```
starrc_shell> report_attribute [get_ports IN*] -application
```

Even if a command accepts a design object name (or list of names) directly, it is good practice to use the get_* commands to create the collection to ensure that the collection contains only items of the specified type.

If object names contain escape characters, use the `-exact` option with the get_* command to specify the names. For example:

```
report_ground_capacitors -of_objects [get_nets -exact {net\\[0\\]}
```

Feedback

The output of some commands is a report. By default, the display scrolls through the entire report. To pause between screens of text (similar to the `more` command in the operating system), set the `sh_enable_page_mode` variable to `true`.

To view a long report in this mode, press the space bar to view each successive screen. To cancel a long report and return to the `starrc_shell` prompt, type the letter *q* .

You can interrupt a command in progress by typing the Ctrl+C key sequence. Computationally intensive commands might take some time to stop. Typing Ctrl+C multiple times terminates the shell and returns to the operating system prompt.

## Parasitic Explorer Variables

Variables hold data. You can control some execution options by specifying the value of application variables. You can also define user variables for convenience in scripts or at the command line. To specify the value of a variable, use the `set` command:

```
starrc_shell> set variable_name value
```

You can use the `set_app_var` command instead of the `set` command when you set the value of an application variable. In this case, if the tool does not recognize the variable name, the tool issues a warning and defines a new user variable with the given name:

```
starrc_shell> set_app_var abc value
Error: Variable 'abc' is not an application variable. Value will still
be set in Tcl. (CMD-104)
Information: Defining new variable 'abc'. (CMD-041)
```

When you set an application variable, the displayed result is the new setting for the variable:

```
starrc_shell> set sh_enable_page_mode true
true
```

If you attempt to set an application variable to an invalid value, the tool issues an error message. For example,

```
starrc_shell> set sh_enable_page_mode maybe
Error: can't set "sh_enable_page_mode": invalid value:
       use true or false
Use error_info for more info. (CMD-013)
```

To determine the current setting for a variable, use the `printvar` command. For example,

```
starrc_shell> printvar sh_enable_page_mode
sh_enable_page_mode  = "false"
```

Feedback

You can use one or more wildcard characters (*) to view a group of variables. For example, to see a list of variables whose names include the string “corner,” enter

```
starrc_shell> printvar *corner*
parasitic_corner_name = ""
```

## Error, Warning, and Information Messages

The Parasitic Explorer tool issues formal messages when a condition arises that requires user attention. Messages have three severity levels:

- Information: No action required if the condition is acceptable
- Warning: Serious condition, likely to be undesirable, but does not stop execution
- Error: Serious condition that prevents analysis from continuing

Some commands provide a `-quiet` option to suppress all warning and error messages. This is common with the `get_*` commands (such as the `get_cells` or `get_nets` commands) because complicated filtering operations might return many unimportant messages while the filter operates on various objects.

## Design Object Attributes

An attribute is a string or value associated with an object in the design that carries some information about that object. For example, the `layer_name` attribute of a parasitic resistor indicates the layer of the resistor shapes. You can write Tcl scripts to get attribute information from the design database and generate custom reports about the design.

Attributes are read-only values that the tool assigns during execution. However, some attributes obtain their values from variables or command options that you specify.

Table 1 lists the commands for working with attributes.

*Table 1 Attribute Commands*

| Attributes | Description |
|---|---|
| `list_attributes` | Lists the names of available attributes by object class. |
| `get_attribute` | Retrieves the value of one attribute associated with one object. |
| `report_attribute` | Displays the values of all attributes associated with one or more objects. |

Feedback

## Listing Attribute Names

The `list_attributes` command displays an alphabetically sorted list of attributes. The list includes the names and properties of the available attributes, but not their values.

**Note:**

Parasitic Explorer does not support user-defined attributes or imported attributes.

To limit the listing to a specific object class, use the `-class` option. You must include the `-application` option. An example of an attribute list is shown here.

```
starrc_shell> list_attributes -class ground_capacitor -application

****************************************
Report : List of Attribute Definitions
...
****************************************
Properties:
    A - Application-defined
    U - User-defined
    I - Importable from design/library (for user-defined)
    S - Settable
    B - Subscripted

Attribute Name        Object             Type     Properties  Constraints
-------------------------------------------------------------------------
capacitance           ground_capacitor   float    A
capacitance_max       ground_capacitor   float    A
capacitance_min       ground_capacitor   float    A
...
```

## Reporting All Attribute Values for an Object

Use the `report_attribute` command to generate a report of attribute values associated with specified objects in the design. You must use the `-application` option. For application attributes that are of the type *collection,* the name of the first object in the collection is displayed. The following example uses the `get_resistors` command to identify parasitic resistors associated with net n833 in a design named Design_A, then passes that result to the `report_attribute` command:

```
starrc_shell> report_attribute [get_resistors -of_objects n833] \
              -application

****************************************
Report : Attribute
...
****************************************

Design     Object      Type          Attribute Name      Value
-----------------------------------------------------------------
```

```
Design_A   resistor    boolean       is_short            false
Design_A   resistor    int           layer_id            2
Design_A   resistor    string        layer_name          metal1
Design_A   resistor    collection    net                 n833
...
```

## Reporting Specific Attribute Values

To report the value of a single attribute for a specific object (or set of objects), use the `get_attribute` command. The following example lists the capacitance values of all of the ground capacitors associated with net n833:

```
starrc_shell> get_attribute [get_ground_capacitors -of_objects n833] \
              capacitance
0.000000 0.000167 0.000023 0.000116 0.000030 0.000082
...
```

# 2

# Using the Parasitic Explorer Tool

You can work with the Parasitic Explorer tool using a Tcl shell or a graphical user interface.

For information about using the Parasitic Explorer tool, see the following topics:

- Creating a GPD for Parasitic Explorer Tool Use
- Using The Interactive StarRC Shell
- Using DSPF Netlist File
- Analyzing and Debugging in Gate-Level Flow
- Analyzing and Debugging in Transistor-Level Flow
- Using Tcl Commands in StarRC Shell

Feedback

# 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.

To use the Parasitic Explorer tool, you must set `PARASITIC_EXPLORER_ENABLE_ANALYSIS: YES` during the extraction to ensure that the GPD contains 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 In_ 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. By default, the StarRC tool does not save detailed information about every short.

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 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 certain properties of parasitic resistors are stored in the GPD during extraction. If you want to examine these attributes with the Parasitic Explorer tool, observe the following guidelines:

- 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

## Using The Interactive StarRC Shell

The following procedure is a general outline of an interactive Parasitic Explorer session.

1. Use the StarRC tool to perform extraction and save parasitics in a GPD.

   You must include the `PARASITIC_EXPLORER_ENABLE_ANALYSIS: YES` command in the extraction command file.

2. Start the Parasitic Explorer tool by entering `starrc_shell` at the operating system prompt.

   ```
   % starrc_shell
   ```

3. If the GPD contains multiple corners, specify the corner name by setting the `parasitic_corner_name` variable:

   ```
   starrc_shell> set parasitic_corner_name corner_name
   ```

4. Read the parasitics from the GPD.

   ```
   starrc_shell> read_parasitics -keep_capacitive_coupling \
                 -format gpd gpd_directory
   ```

5. Specify the current design, which is the name used in the `BLOCK` command in the StarRC command file that is used for extraction.

   ```
   starrc_shell> current_design design_name
   ```

6. Use Parasitic Explorer commands to find the parasitics associated with design objects.

   ```
   starrc_shell> get_coupling_capacitors ...
   starrc_shell> get_ground_capacitors ...
   starrc_shell> get_resistors ...
   ```

7. Use Tcl commands to examine the attributes of the parasitics.

   ```
   starrc_shell> report_attribute ...
   ```

8. Use Tcl commands to perform general functions such as storing parasitics into user variables, operating on those variables, and writing data into a custom report.

   ```
   starrc_shell> set aggr_cap ...
   starrc_shell> set new_cap [expr $aggr_cap ...
   starrc_shell> echo ...
   starrc_shell> puts ...
   ```

9. End the session with either of the following commands:

   ```
   starrc_shell> quit
   starrc_shell> exit
   ```

You can also create Tcl scripts to carry out complex or repetitive tasks.

## Application Examples

These commands are examples of how to work with parasitic objects retrieved from a GPD and are not necessarily complete Tcl scripts.

### Example 1

The following Tcl code finds wire segments with width less than 5 nm.

```
foreach_in_collection net [get_nets *] {
   foreach_in_collection res [get_resistors -of_objects $net]  {
      if { [get_attribute $res width] < 0.005 } {
         puts [format "Net:%s ResNodes:%d-%d Width:%g" \
            [get_attribute $net full_name] \
            [get_attribute $res node1_index] \
            [get_attribute $res node2_index] \
         ]
      }
   }
}
```

The output appears as follows:

```
Net:net1 ResNodes:1-2 Width:0.002
Net:net13 ResNodes:43-32 Width:0.0045
Net:net99 ResNodes:23-25 Width:0.001
```

### Example 2

The following Tcl code finds the total net wire length by layer. Assume that variable $net is already set as in Example 1.

```
array set netLen {}
foreach_in_collection res [get_resistors -of_objects $net] {
   set res_lyr [get_attribute $res layer_name]
   set res_len [get_attribute $res length]
   if {[info exists netLen($res_lyr)]} {
     set $netLen($res_lyr) [expr {$res_len + $netLen($res_lyr)}]
   } else {
     set netLen($res_lyr) $res_len
   }
}
foreach key [array names netLen] {
   if {$netLen($key) > 0} {
      puts [format "(%s %g)" $key $netLen($key)]
   }
}
```

Feedback

The output appears as follows:

```
(metal2 1.375)
(metal3 3.76)
(metal4 9.205)
```

**Example 3**

The following Tcl code finds the top 100 nets with the largest ratio of ground capacitance between parasitic corners.

```
array set gcap_ratio {}
foreach_in_collection net [get_nets *] {
   set gcap1 0
   set gcap2 0
   foreach_in_collection gcap [get_ground_capacitors -of_objects $net \
                               -parasitic_corners "cworst cbest"] {
      set gcap1 [expr $gcap1 + [lindex [get_attribute $gcap \
          capacitance] 0]]
      set gcap2 [expr $gcap2 + [lindex [get_attribute $gcap \
          capacitance] 1]]
   }
   set gcap_ratio([get_attribute $net name]) [expr $gcap1/$gcap2]
}

set cntr 0
foreach {net_name gcap_ratio} [eval {lsort -stride 2 -real -index 1 \
                           -decreasing [array get gcap_ratio]}] {
   puts "Net:$net_name Ratio:$gcap_ratio"
   incr cntr
   if {$cntr >= 100} {
      break
   }
}
```

The output appears as follows:

```
Net:net95 Ratio:122.875
Net:net284 Ratio:118.502
Net:net105 Ratio:91.18
...
```

## Using DSPF Netlist File

You can specify Detailed Standard Parasitic Format (DSPF) netlist files with the `read_parasitics -format dspf file_name` command to use the Parasitic Explorer commands for GPD annotations in the SPF flow.

Feedback

1. Start the Parasitic Explorer tool by entering `starrc_shell` at the operating system prompt.

```
% starrc_shell
```

2. Read the parasitics from the .spf file.

```
starrc_shell> read_parasitics -keep_capacitive_coupling \
              -format dspf design.debug.spf
```

3. Specify the current design, which is the name used in the `BLOCK` command in the StarRC command file that is used for extraction.

```
starrc_shell> current_design design_name
```

4. Use Parasitic Explorer commands to find the parasitics associated with design objects.

```
starrc_shell> get_coupling_capacitors ...
starrc_shell> get_ground_capacitors ...
starrc_shell> get_resistors ...
```

5. End the session with either of the following commands:

```
starrc_shell> quit
starrc_shell> exit
```

Feedback

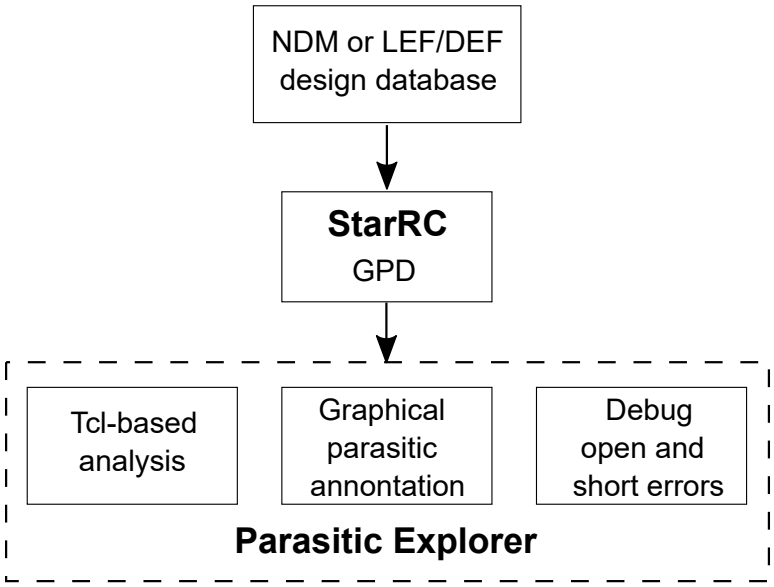
# Analyzing and Debugging in Gate-Level Flow

You can analyze and debug RC elements for selected nets in the parasitic explorer gate-level flow.

In the gate-level flow, the Parasitic Explorer tool

- Provides an environment for advanced analysis of parasitics
- Supports the Tcl language with Synopsys Tcl extensions
- Provides a graphical environment to annotate parasitics and to debug open and short errors
- Uses the `starrc_shell` command

*Figure 2 Parasitic Explorer Gate-Level Flow*




For information to analyze and debug parasitics, see the following topics:

- Setting Up the Gate-Level Flow
- Displaying Parasitic Elements in a Layout View
- Viewing Open and Short Errors With the Error Browser GUI
- Managing Open and Short Errors Using Summary View
- Analyzing Open and Short Errors
- Reporting Power Net Names in Short Summary File

Feedback

## Setting Up the Gate-Level Flow

To setup a gate-level flow,

1. Run extraction using the following command:

   ```
   PARASITIC_EXPLORER_ENABLE_ANALYSIS: YES
   ```

2. Start the Parasitic Explorer tool by invoking the StarRC shell:

   ```
   % starrc_shell
   ```

3. Source the starrc_shell_init.tcl file to read the parasitics from the GPD and specify the current design:

   ```
   starrc_shell> source <gpd_directory>/starrc_shell_init.tcl
   ```

*Example 1 Commands in the starrc_shell_init.tcl File*

```
# Reads the parasitics
set gpd_read_remove_buslike_escape false
read_parasitics -keep_capacitive_coupling -format GPD <gpd_directory>

# Specifies the current design
current_design <design_name>
```

4. Source the starrc_shell_load_layout.tcl file to read the physical design database and check the physical database for consistency:

   ```
   starrc_shell> source <gpd_directory>/starrc_shell_load_layout.tcl
   ```

*Example 2 Commands in the starrc_shell_load_layout.tcl File*

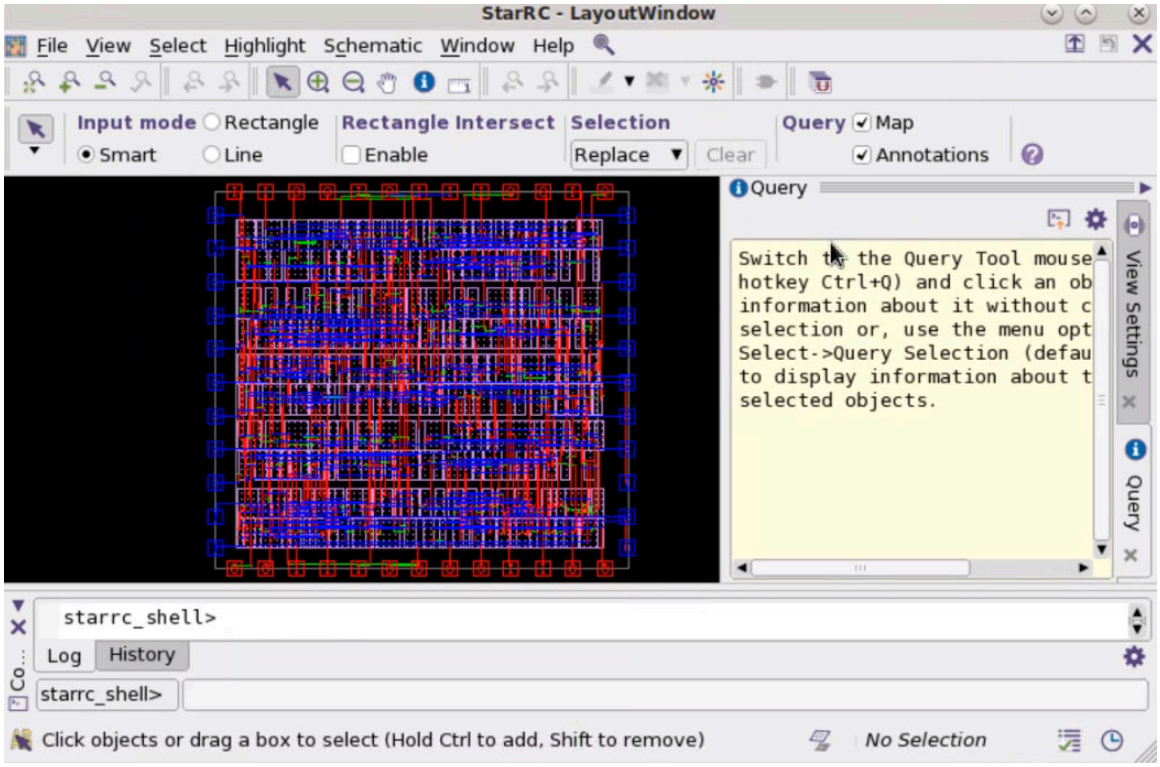
```
# Reads a design database
set_layout_database_options -physical_enable_clock_data  \
            -physical_lib_path {design_library_files}    \
            -physical_design_path {design_physicaldata_files}

# Checks the physical database for consistency
check_layout_database
```

5. Invoke the GUI. The StarRC - Layout window appears (Figure 3). The original terminal screen is still accessible.

Feedback

```
starrc_shell> start_gui
```

*Figure 3 StarRC Parasitic Explorer GUI Layout Window for Gate-Level Flow*



Feedback

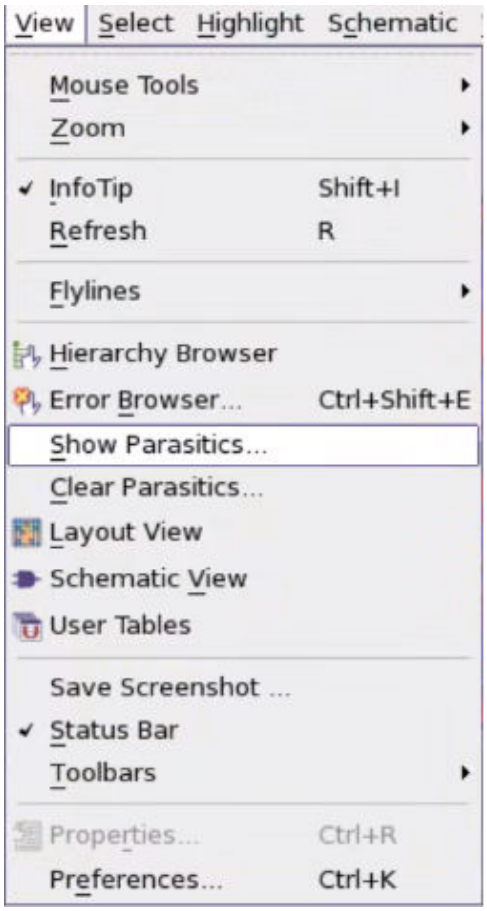
## Displaying Parasitic Elements in a Layout View

For a gate-level flow, you can use a GUI to visualize the RC elements associated with selected nets in an NDM or LEF/DEF design database.

To display parasitic elements in a layout view,

1. Set up the gate-level flow (see Setting Up the Gate-Level Flow and Figure 3).
2. Click **View > Show Parasitics** (Figure 4).

*Figure 4 Show Parasitics*

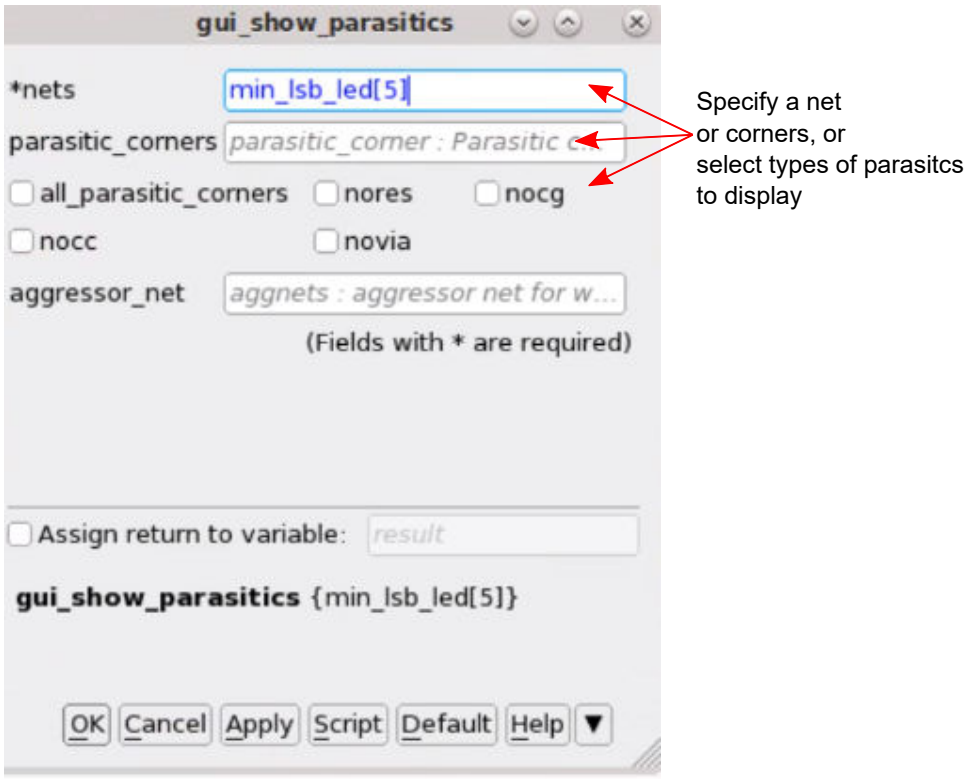



The gui_show_parasitics window appears (Figure 5).

3. Enter a net name and specify corners or select parasitics to display as needed (Figure 5).

Feedback

*Figure 5 The gui_show_parasitics Window*

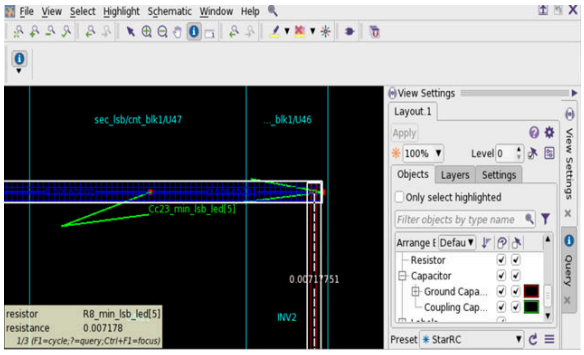



When you use the command line (Figure 48) to run `gui_show_parasitics` command, you can use options of the `gui_show_parasitics` command to restrict the displayed parasitics. For example, you can select a specific net, specify which corners to use, and disable the display of certain types of parasitics. See Chapter 4, Parasitic Explorer Command Reference for more information about the command.

4. Click **Apply** (Figure 5) to view the specified net (Figure 6 and Figure 10).

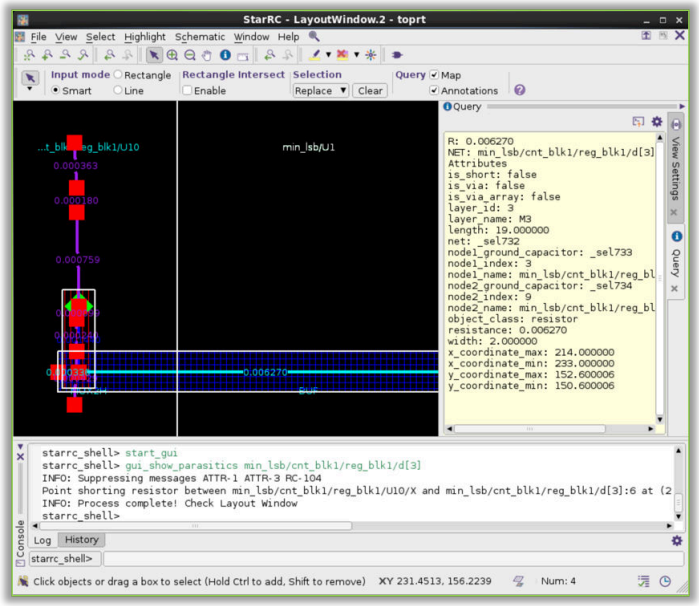
Feedback

*Figure 6 Specified Net is Highlighted*




5. Zoom in to the location of the net to view the annotated RC elements by using the Zoom tool (in the View menu) or the + keyboard shortcut. Flylines represent resistors, squares represent ground capacitors, and diamonds represent pin capacitors (Figure 7).

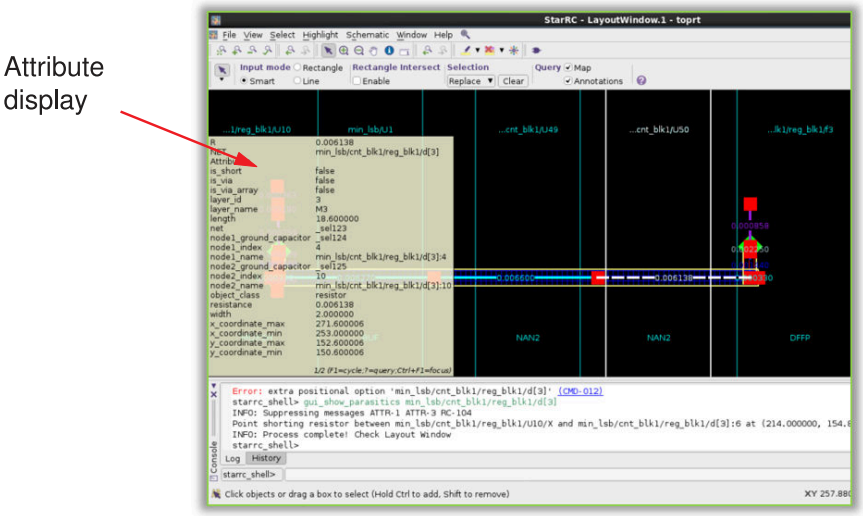
*Figure 7 Annotated Parasitics*




6. Hover the pointer over a parasitic element to display the element attributes (Figure 8).

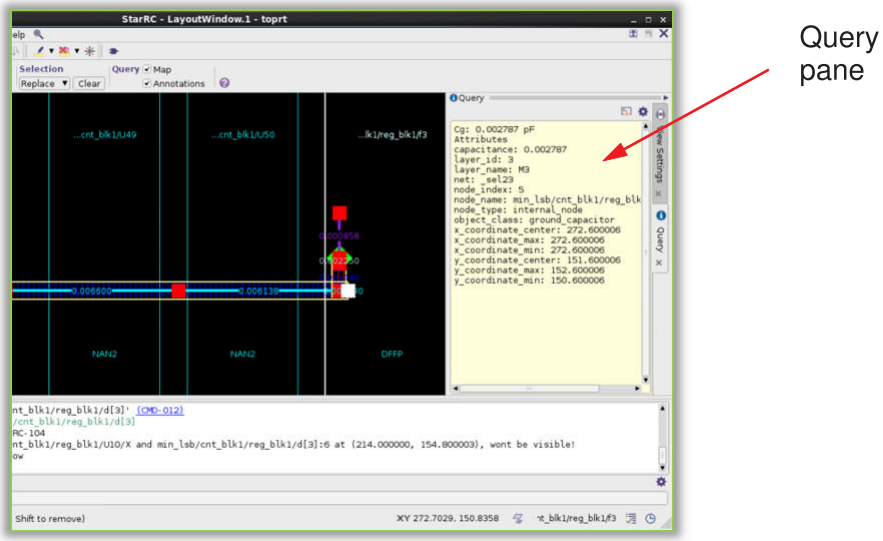
Feedback

*Figure 8 Annotated Parasitics With Attribute Display*




7. Left-click on an element to populate the **Query** pane with the element attributes (Figure 9).

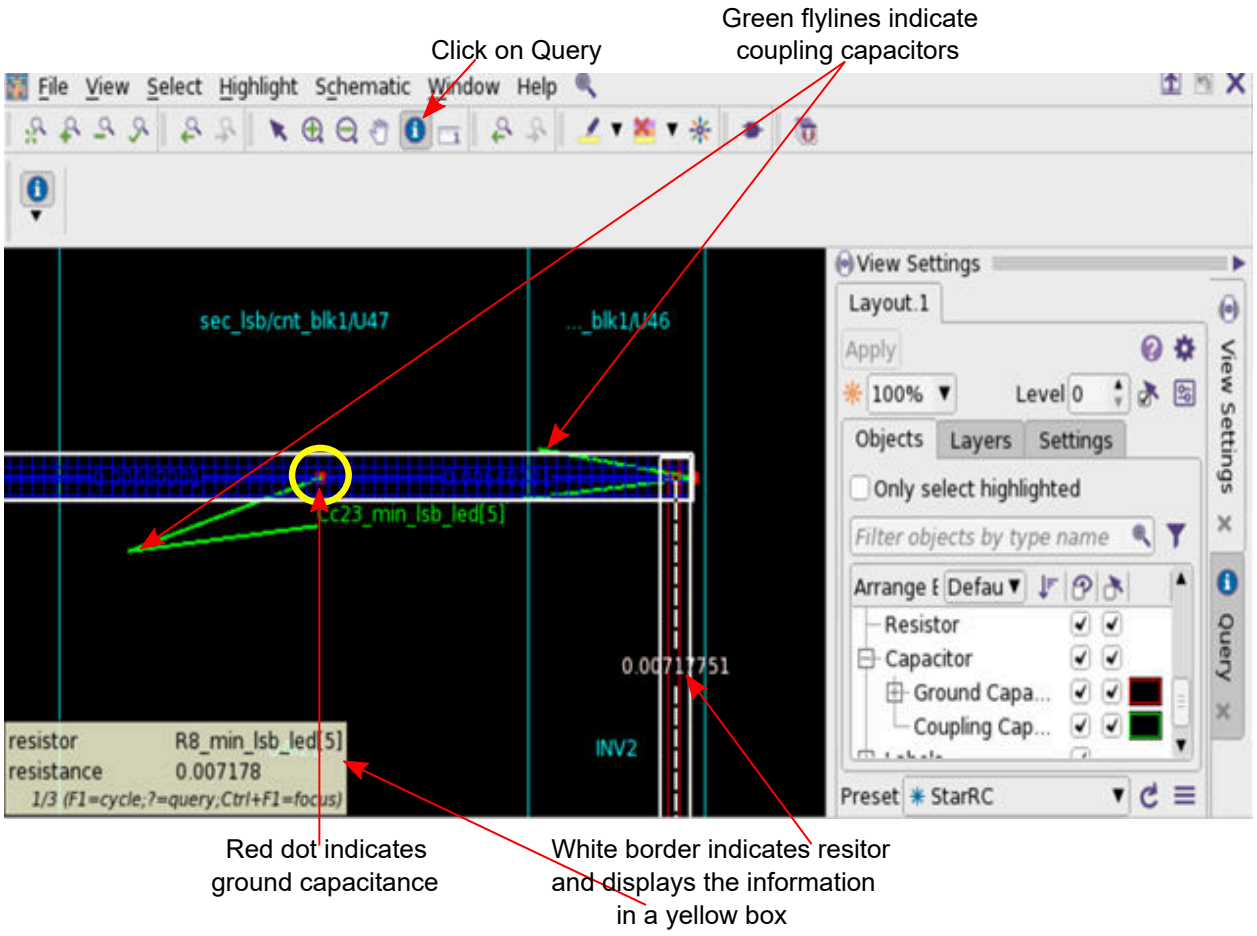
*Figure 9 Annotated Parasitics With Attributes in the Query Pane*



Feedback

8. Use the **Query** icon (Figure 10) to query resistance, ground capacitance, and coupling capacitance and view design and net parasitics after choosing **Show Parasitics** (Figure 4). Also, click on the following tabs to select and deselect check box to view appropriate types of parasitics for the specified net:
   - **Query**: Displays information of resistor and capacitor with ground capacitance and coupling capacitance.
   - **View Settings**: Displays layer and setting information.

*Figure 10 Query Icon, and View Settings and Query Tabs*




9. Clear the parasitics with the `gui_clear_parasitics` command.

```
starrc_shell> gui_clear_parasitics
```

10. When you are done examining the parasitic elements, close the GUI window.

11. Exit the StarRC shell session with the `quit` or `exit` command.

```
starrc_shell> quit
```

Feedback

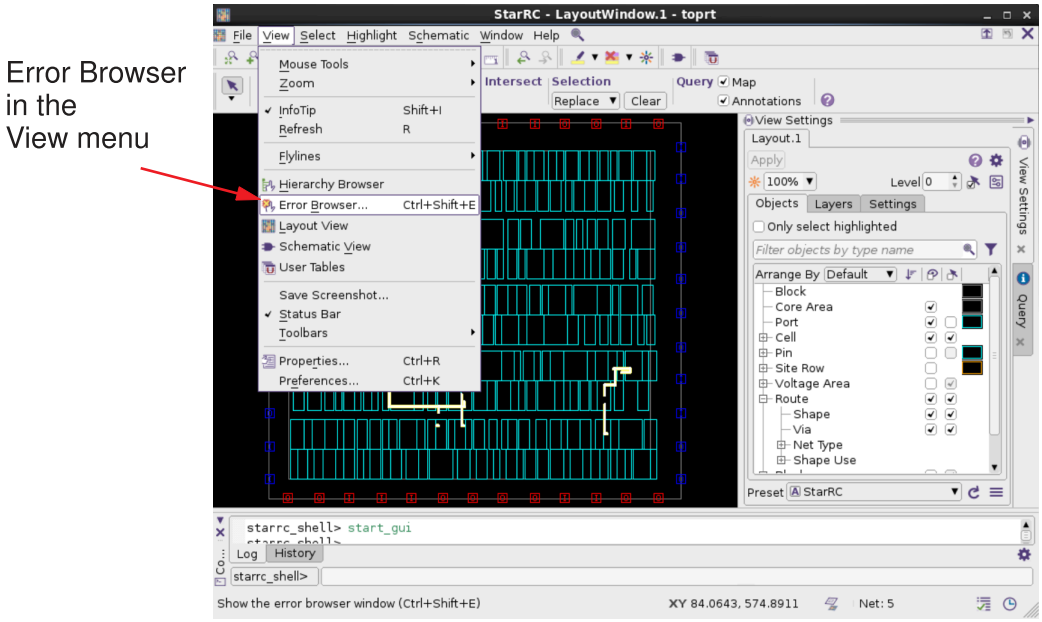
## Viewing Open and Short Errors With the Error Browser GUI

For a gate-level flow, you can use the Parasitic Explorer error browser to examine opens and shorts found by the StarRC tool during extraction.

The general procedure for using the error browser GUI is as follows:

1. Set up the gate-level flow (see Setting Up the Gate-Level Flow and Figure 3).
2. Choose **View > Error Browser** (Figure 11).

*Figure 11 Error Browser Selection*




3. Choose **File > Read Error File** (Figure 12).

   The **Error Browser** dialog box appears. Select an error file; the default name is starrc_openshort.err. A list of nets with opens and shorts appears in the upper pane (Figure 13).

Feedback

*Figure 12 Reading an Error File*

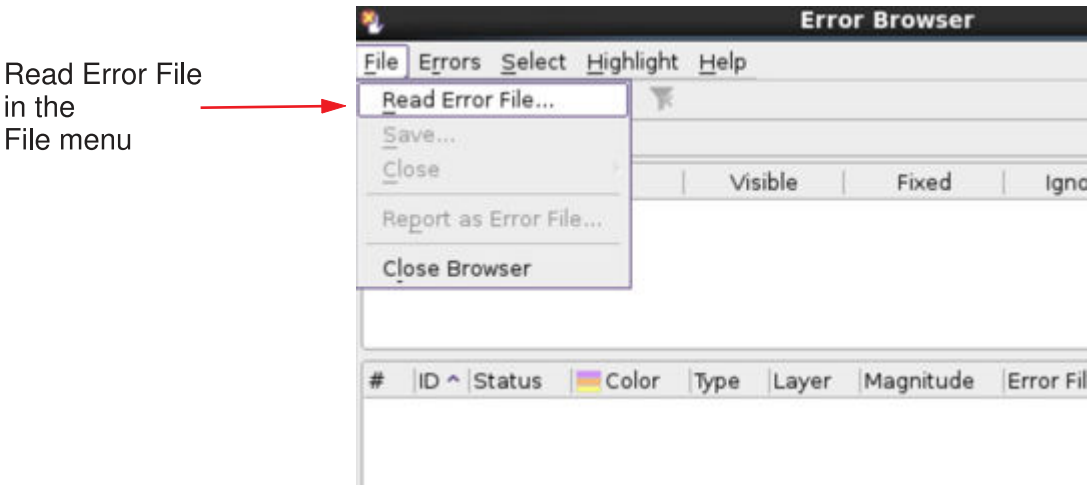
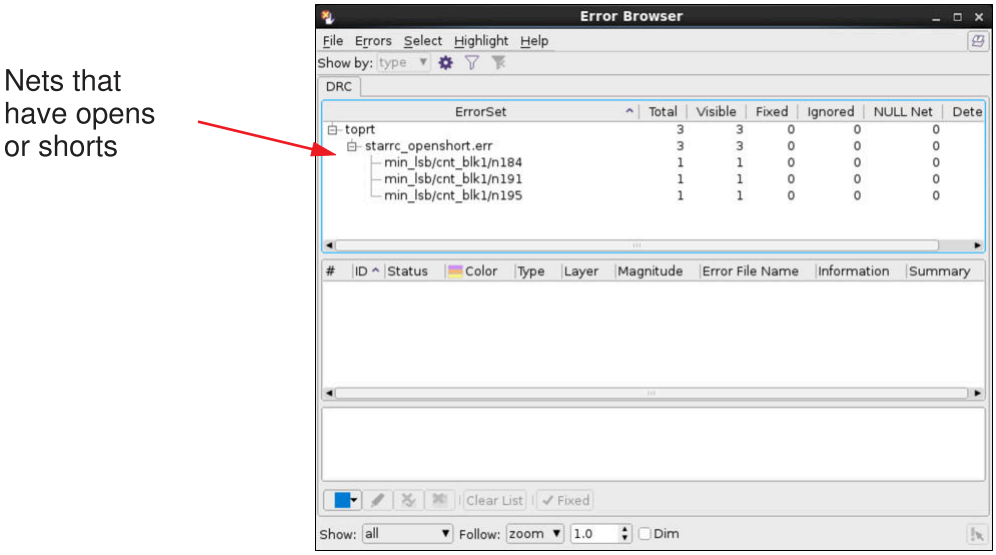



*Figure 13 List of Nets With Opens and Shorts*

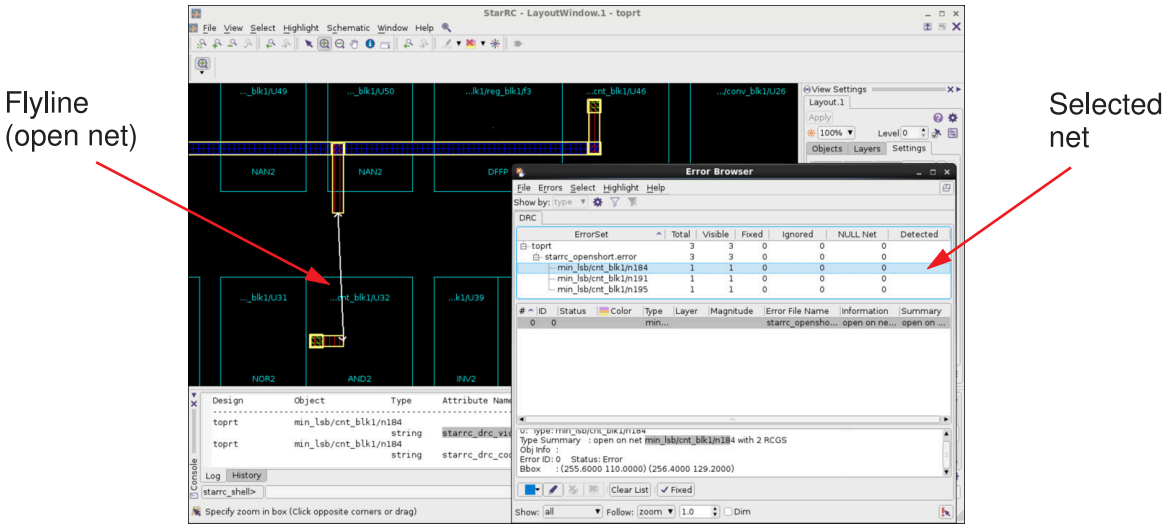


Feedback

4. Select a net from the list in the **Error Browser** dialog box and click **Apply**.

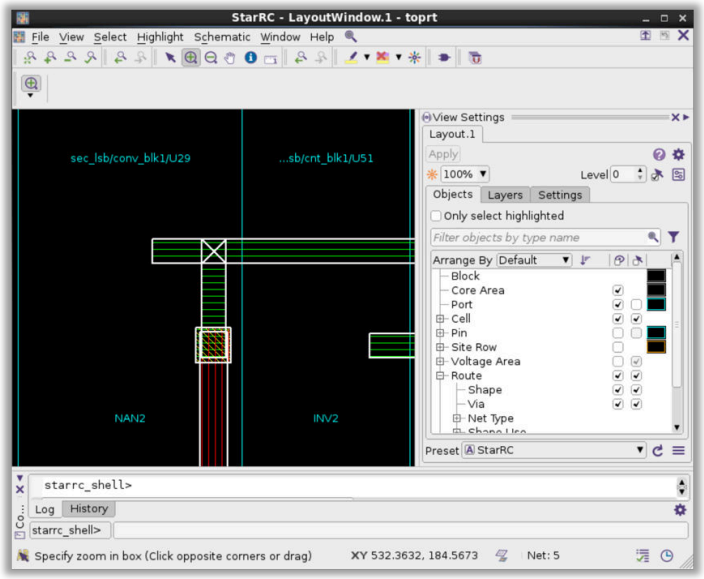
   The selected net is displayed (Figure 14 for an open net. A flyline indicates the location of the open error.

*Figure 14 Open Net Display*




A shorted net appears (Figure 15).

*Figure 15 Shorted Net Display*

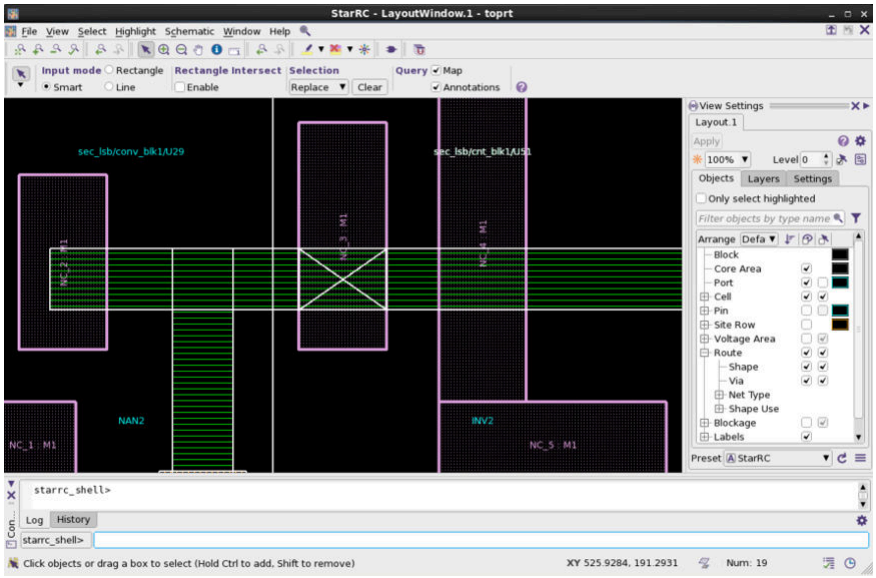


Feedback

5. To examine and debug shorts, select a shorted net from the error browser. An X appears on the layout at the location of the short. If a net is shorted in multiple locations, each short is listed in the error browser. You can navigate through the shorts by clicking on them in the error browser.
6. You can also select a net by name. In the layout window, choose **Select > By Name**. In the **Select by Name** dialog box, select the design object type and enter a name in the **Name** field.
7. Display the noncritical material in the region immediately surrounding the short (Figure 16 by using the following command:

   ```
   starrc_shell> gui_show_short_regions -gpd <gpd_dir>
   ```

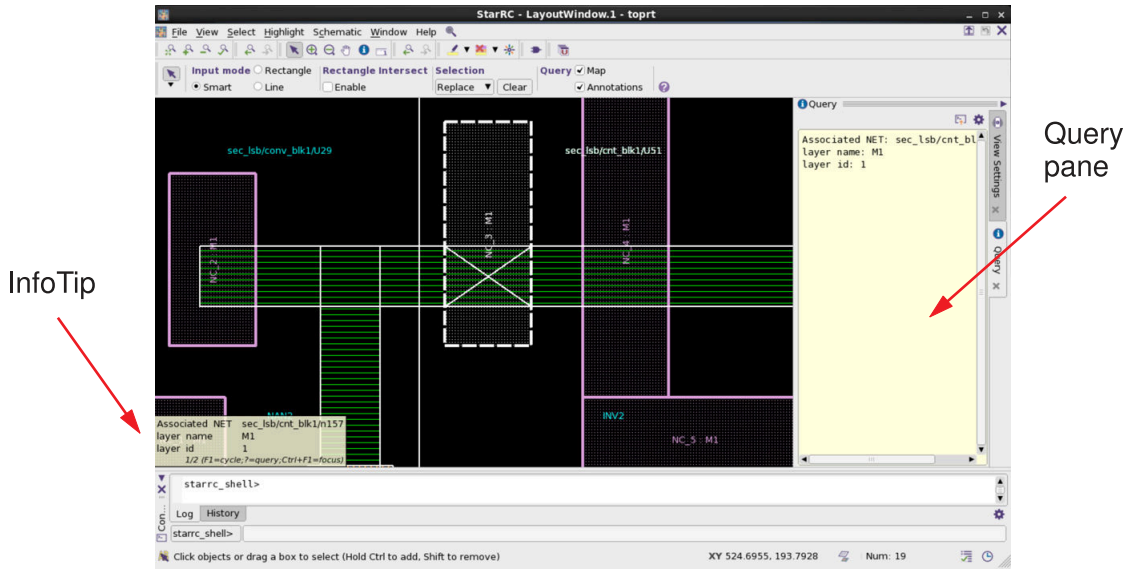
*Figure 16 Shorted Net Display With Nearby Noncritical Material*




You can view the net name and layer information of every shape in the short region by clicking on the object and looking at the InfoTip or the **Query** pane (Figure 17).

Feedback

*Figure 17 Shorted Net Display With Query Information*




For more examples to view open and short errors using Tcl command, see starrc_gpd_read_opens_shorts.

8. When you are done examining the nets, close the GUI window.
9. Exit the StarRC shell session with the `quit` or `exit` command.

```
starrc_shell> quit
```

### See Also

- Analyzing Open and Short Errors
- Managing Open and Short Errors Using Summary View
- starrc_gpd_read_opens_shorts

Feedback

## Managing Open and Short Errors Using Summary View

When you create a GPD with the `PARASITIC_EXPLORER_ENABLE_ANALYSIS: YES` command, the tool generates the following files:

- shorts_all.sum: Generated by the StarRC extraction tool where the shorts types are categorized.
- starrc_shell_error_summary_view.tcl: Automatically generates the Tcl file to view the heat map of all errors, including open and short errors.

When you source the Tcl file using the following command, the tool reads the physical data from LEF/DEF or NDM design for the GUI along with GPD parasitics and then opens the GUI and displays the summary view (Figure 18).

```
starrc_shell> source <gpd_directory>/my_summary_view.tcl
```

*Example 3 Commands in a Tcl File to Read a Design Database*

```
# Reads the parasitics
set gpd_read_remove_buslike_escape false
read_parasitics -keep_capacitive_coupling -format GPD <gpd_directory>

# Specifies the current design
current_design <design_name> -only_link_in_pe

# Reads a design database
set_layout_database_options -physical_enable_clock_data  \
            -physical_lib_path {design_library_files}    \
            -physical_design_path {design_physicaldata_files}

# Checks the physical database for consistency
check_layout_database

start_gui

# Reads the opens and shorts information to display the summary view
starrc_gpd_read_opens_shorts -gpd <gpd_dir> -summary_view
```

Feedback

*Figure 18 Summary View Shows Layer and X Markers in Distinct Colors*

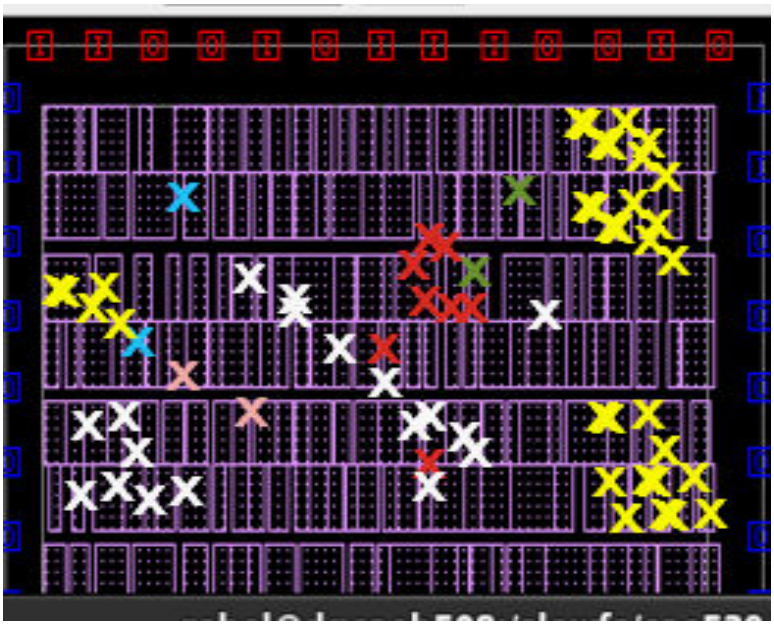


Table 2 lists the shorts error types and the respective color of X markers to categorize and prioritize shorts and open errors. Figure 18 shows X markers in the summary view.

*Table 2 Shorts and Open Error Types With Color of X Markers*

| Error type | Color |
|---|---|
| Short to net | Red |
| Short to unselected net | Orange |
| Short to unselected net (power nets) | Yellow |
| Short to skip cell | Green |
| Short to fill | Cyan |
| Short to blockage | Pink |
| Open error | White |

For more examples to view open and short errors using Tcl command, see starrc_gpd_read_opens_shorts.

Feedback

**See Also**

- Analyzing Open and Short Errors
- Viewing Open and Short Errors With the Error Browser GUI
- Reporting Power Net Names in Short Summary File

## Analyzing Open and Short Errors

To analyze open and short errors of a large design,

- Generate a heat map by sourcing the starrc_shell_error_summary_view.tcl file to display in the summary view that helps to
  - Quickly view all shorts and opens error
  - Identify areas showing many errors
  - Focus on errors with the `-type`, `-short_types`, or `-window` option
  - Categorize and prioritize shorts errors with distinct color of X markers for each type of shorts error, as shown in Table 2
- Generate an error file with the `starrc_gpd_read_opens_shorts` command, as shown in Example 3, that helps to
  - Sort shorts and opens errors
  - Focus on shorts and opens errors with the `-type`, `-short_types`, or `-window` option
  - Narrow down the selected types of shorts to debug using the `-short_types` option

*Example 4 Generating Error file (.err) to Use in the Error Browser GUI*

```
starrc_shell> starrc_gpd_read_opens_shorts -gpd my.gpd -type short \
        -window 55,1634,1825,2175 -short_types (net unselectable \
                              nonselected skip_cell fill blockage)
        -error_file my_wrapper.err

**************************************
Report : Error counts
**************************************
Short errors : 19292
  short to net              : 12079
  short to fill             : 358
  short to blockage         : 6816
  short to unselectable net : 39

starrc_shell> ls -lh my_wrapper.err
3.8G my_wrapper.err
```

Feedback

**See Also**

- Managing Open and Short Errors Using Summary View
- Viewing Open and Short Errors With the Error Browser GUI
- starrc_gpd_read_opens_shorts

## Reporting Power Net Names in Short Summary File

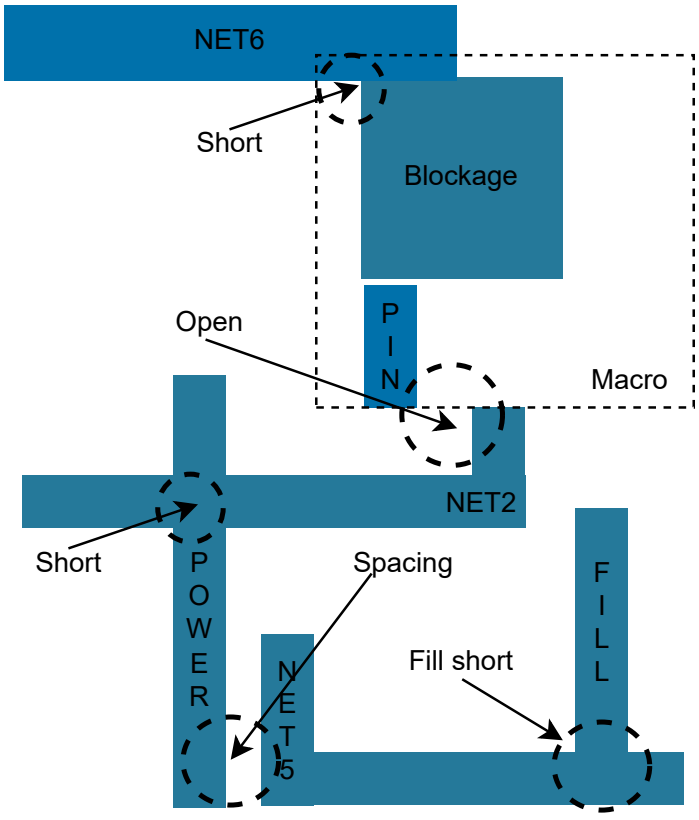
The Parasitic Explorer tool reports shorts from *extracted signal nets to a non-extracted power net*, even if you have set the `POWER_EXTRACT` command to `NO`. To generate this report, you need to set the `ENHANCED_SHORT_REPORTING` command to either `YES` or `COMPLETE`.

The tool reports power net names in the following format:

```
Short between net {net name} and power net {power net name} Layer = {}
 BBox={}
```

Example 5 shows a portion of a report for the net structure shown in Figure 19.

*Figure 19 Identifies Power Nets Between Short NET4 and Open NET1*



*Example 5 Reports Shorts From Extracted Signal Nets to Non-Extracted Power Net*

```
Short between NET6 and power net vss Layer=M6 Bbox=(447.052,436.477), \
(447.097,436.477)
Open between NET2 and power net vss Layer=M6 Bbox=(447.052,436.477), \
(447.097,436.477)
```

Feedback

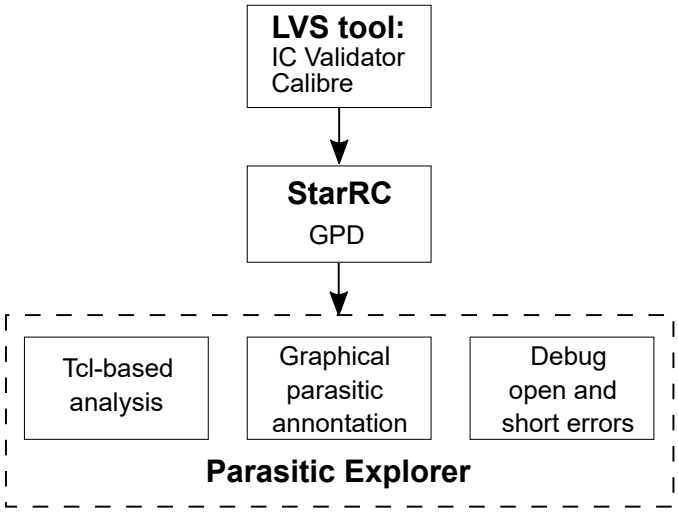
# Analyzing and Debugging in Transistor-Level Flow

You can view, analyze, and debug parasitics and open and short errors for selected nets in the parasitic explorer transistor-level flow.

In the transistor-level flow, the Parasitic Explorer tool

- Provides an environment for advanced analysis of parasitics for gate-level and transistor-level extraction flows
- Supports the Tcl language with Synopsys Tcl extensions
- Provides a graphical environment to annotate parasitics and to debug open and short errors

*Figure 20 Parasitic Explorer Transistor-Level Flow*




For information to analyze and debug parasitics, see the following topics:

- Accessing the Interoperable Process Design Kit (iPDK)
- Setting Up the Transistor-Level Flow
- Loading and Analyzing GPD Parasitics
- Viewing and Analyzing Open and Short Errors
- Analyzing Parasitics Using StarRC Virtuoso Integration
- Viewing the Heatmap Report

## Accessing the Interoperable Process Design Kit (iPDK)

For a transistor-level extraction flow, you need the iPDK to create and setup OpenAccess (OA) libraries and the lib.def file. The iPDK includes the following information to create schematics and layout for a design:

- Parameterized cells (PCell) for layout instantiation of circuit devices
- Symbols for circuit design and schematic creation
- Callbacks to calculate device parameters
- Technology files to define design rules, connectivity information, and layers to use in the layout
- Additional information to enable advanced features based on process nodes and user requirements

For information to access and install the iPDK, contact your vendor or Synopsys support.

## Defining Libraries for an OpenAccess View

To define libraries using iPDK,

1. Install the iPDK provided by your vendor.
2. Copy the cds.lib file into the lib.defs file, as shown by the following command:

   ```
   cp cds.libs lib.defs
   ```

   **Note:**

   Save the lib.defs file in your working directory.

For detailed information about iPDK and setting up the lib.def and technology files, see the Custom Compiler documentation on SolvNetPlus.

Feedback

## Setting Up the Transistor-Level Flow

For a transistor-level parasitic explorer flow, you need both GPD and OpenAccess (OA) view.

The following general procedure is as follows:

1. List the commands in the StarRC command file as shown in Example 6 to create both GPD and an OA view in one run.

*Example 6 Creating OpenAccess View*

```
# Creates and saves a GPD
REDUCTION:NO
XREF:YES
EXTRA_GEOMETRY_INFO: NODE RES
NETLIST_TAIL_COMMENTS: YES
PARASITIC_EXPLORER_ENABLE_ANALYSIS: YES

# Creates an OpenAccess view
OA_LIB_DEF: TECHLIB/lib.defs
OA_LIB_NAME: my_library_OA
OA_CELL_NAME: TOP_CEL
OA_VIEW_NAME: starrc_physical_view
OA_PHYSICAL_ONLY_VIEW: YES
NETLIST_FORMAT: OA
```

2. Start the Parasitic Explorer tool by invoking the StarRC shell:

```
% starrc_shell
```

3. Source the starrc_shell_init.tcl file to read the parasitics from the GPD and specify the current design:

```
starrc_shell> source <GPD_DIR>/starrc_shell_init.tcl
```

*Example 7 Tcl File to Read GPD and Specify Current Design*

```
# Commands in *.tcl file
set gpd_read_remove_buslike_escape false
read_parasitics -keep_capacitive_coupling -format GPD <GPD_DIR>

current_design <design_name>
```

4. Set the `SYNOPSYS_FEATURE_GPD_OPEN_SHORT` environment variable to `1`:

```
setenv SYNOPSYS_FEATURE_GPD_OPEN_SHORT 1
```

**Note:**

Set the environment variable before you use the `starrc_explorer &` command. Otherwise, the GUI might not display the menus correctly.

Feedback

5. Set the existing Custom Compiler shell (custom_shell) at the Unix path as shown in the following example:

   ```
   % set path = (/global/apps/customcompiler_2020.12-SP1/bin $path)
   ```

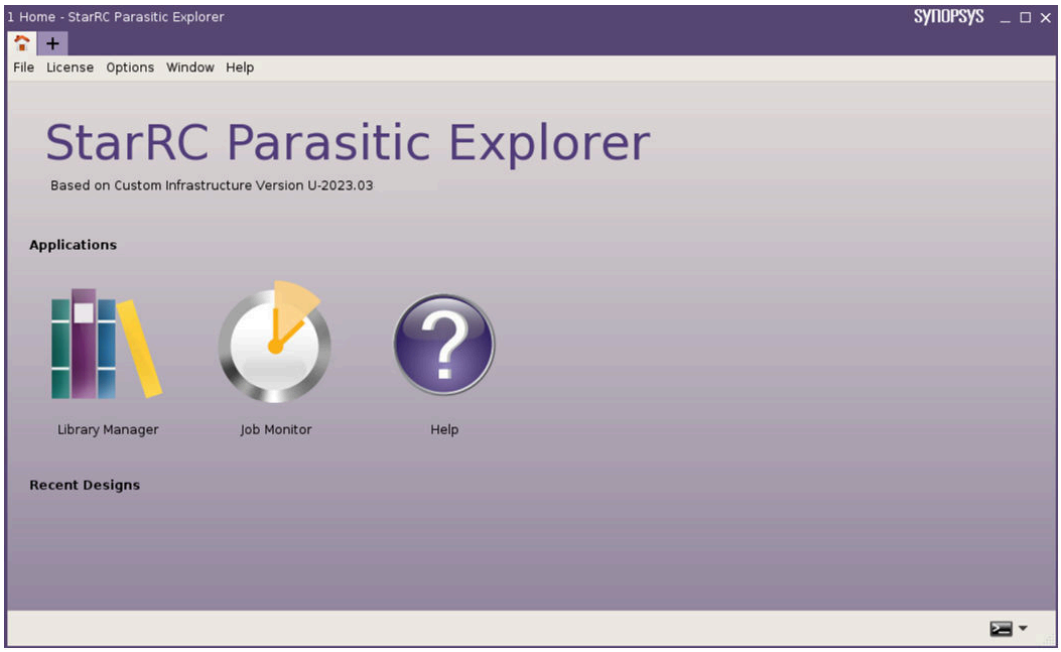
   Or

   ```
   % module load customcompiler
   ```

6. Start StarRC Parasitic Explorer using the OA view:

   ```
   % starrc_explorer &
   ```

*Figure 21 StarRC Parasitic Explorer GUI Window for Transistor-Level Flow*




7. Click Library Manager to open the Layout Editor window.

Feedback

## Loading and Analyzing GPD Parasitics

To view, highlight, and query resistance, coupling ground, and coupling capacitance and to analyze the uploaded GPD parasitics for a specific net:

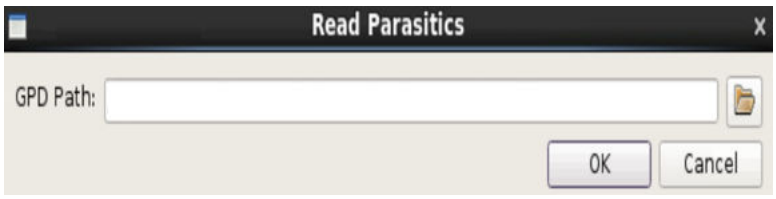
1. Start the GUI and click Library Manager to open the Layout Editor window (see Setting Up the Transistor-Level Flow and Figure 21).
2. Click **Parasitics > Read Parasitics**.

   The Read Parasitics dialog box appears (Figure 23).

*Figure 22 Read Parasitics Menu*

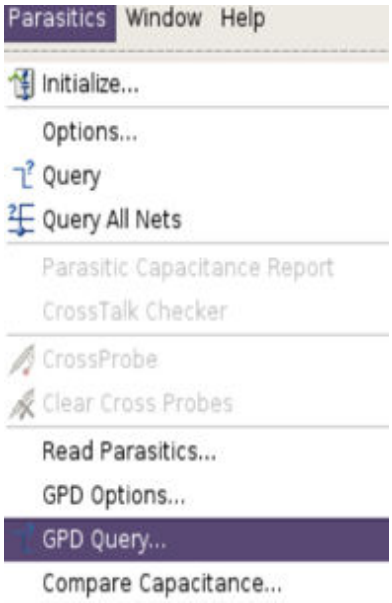

3. Select a GPD directory from your local folder in the **GPD Path** box (Figure 23).

*Figure 23 Read Parasitics Dialog Box*



4. Click **OK** to upload the selected GPD directory.
5. Click **Parasitics > GPD Query...** (Figure 24).

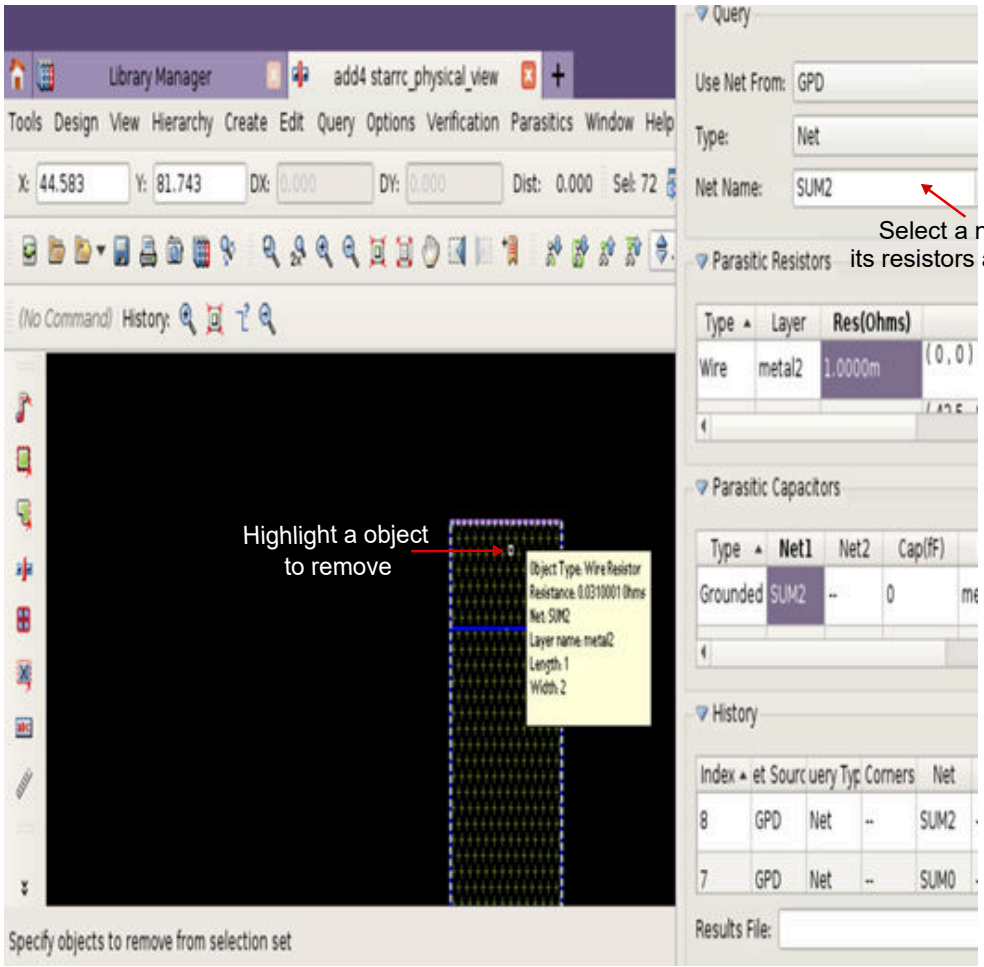
*Figure 24 GPD Query Menu*




6. Select a net from the list to display all resistors and capacitors and highlight a resistor or capacitor segment to analyze RC elements (Figure 25).

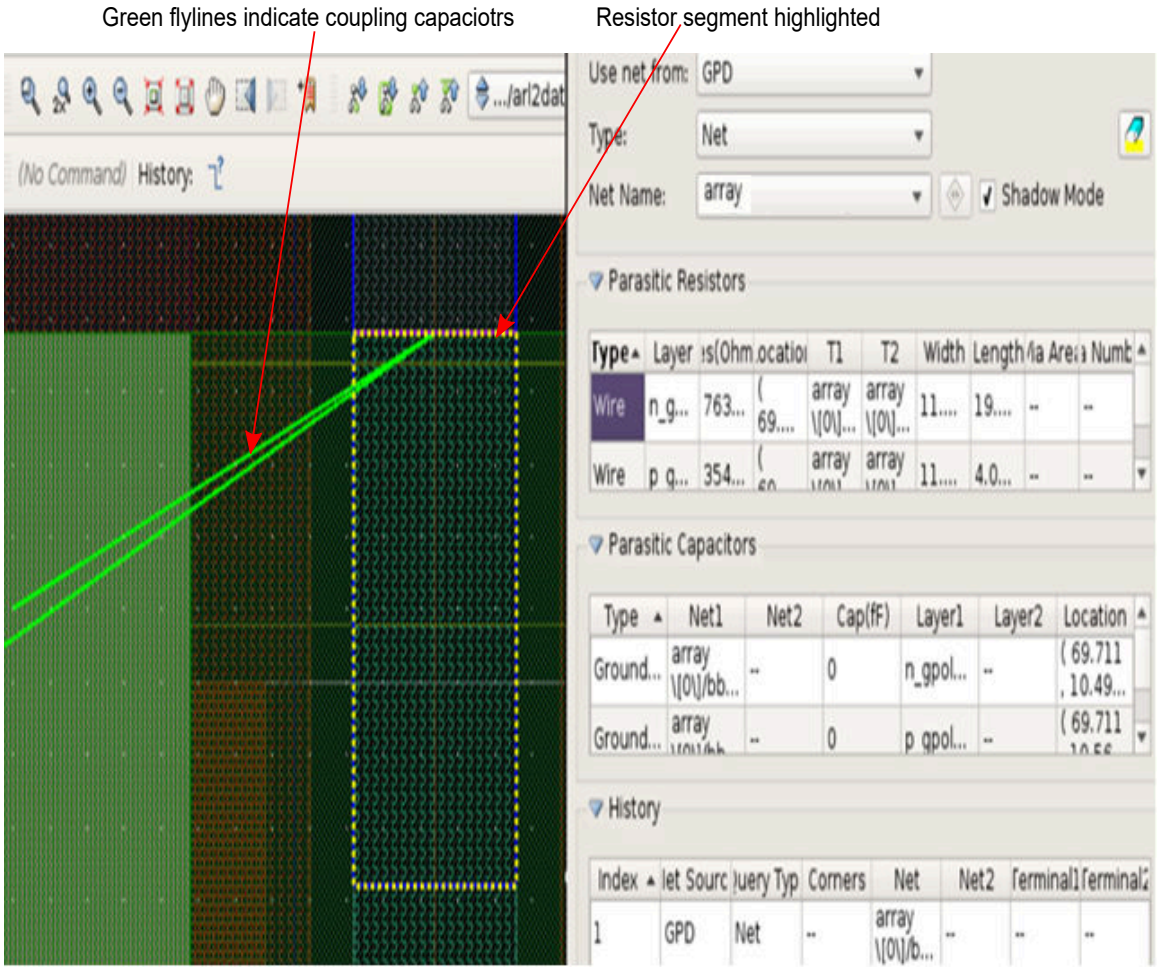
Feedback

*Figure 25 Highlighting Resistor Segment*



Feedback

*Figure 26 Green Flylines Indicate Coupling Capacitors*



Feedback

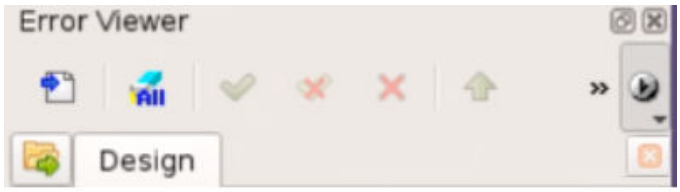
## Viewing and Analyzing Open and Short Errors

To analyze view, highlight, and analyze open and short errors for a specific net found by the StarRC tool during extraction:

1. Start the GUI and click Library Manager to open the Layout Editor window (see Setting Up the Transistor-Level Flow and Figure 21).
2. Click **Windows > Assistants > Error Viewer**.

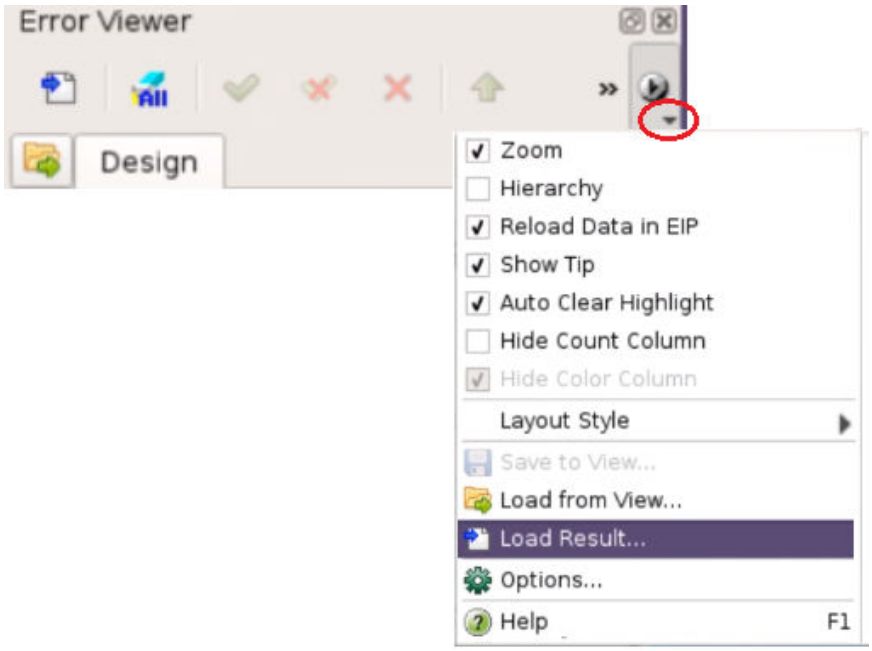
   The Error Viewer window appears (Figure 27).

*Figure 27 Error Viewer Window*



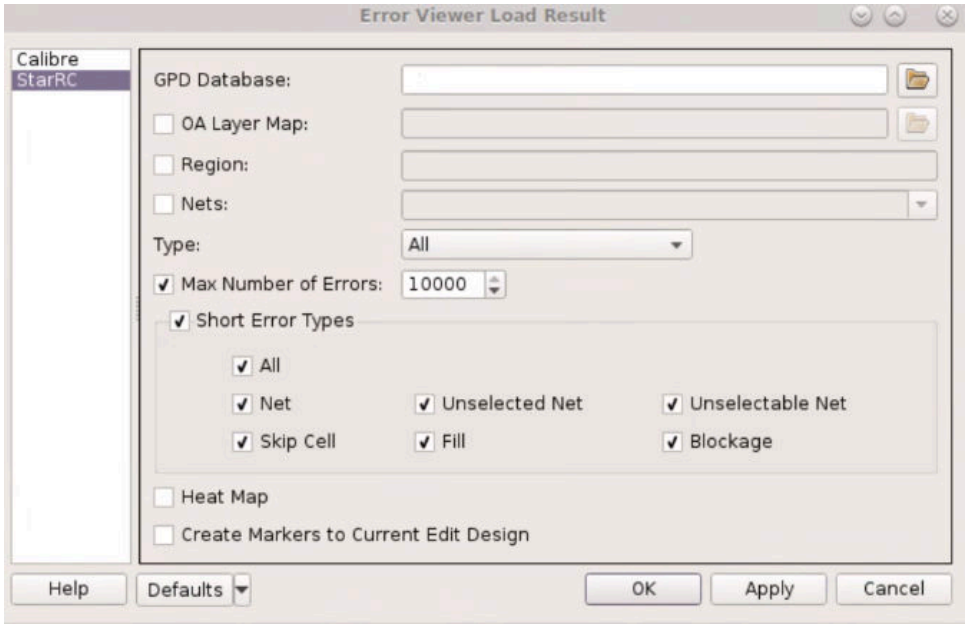

3. Click drop-down key **> Load Result...** (Figure 28).

*Figure 28 Error View Load Result Menu*




The Error Viewer Load Result window appears (Figure 29).

*Figure 29 Error Viewer Load Result Window*




4. In the Error Viewer Load Result window (Figure 29),
   a. Select StarRC.
   b. Select a GPD directory from your local folder in the **GPD Database** box.
   c. Select **Short Error Types**.
   d. Click **Apply** and **OK**.

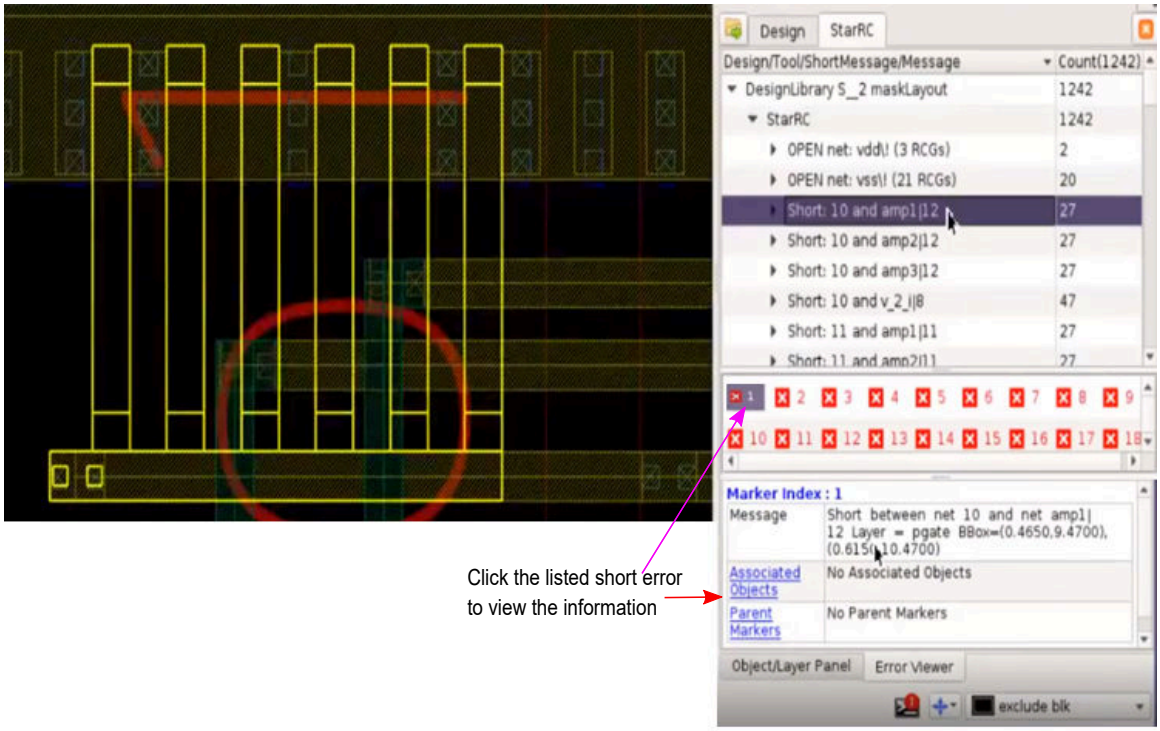
   The Layout Editor window displays all open and short errors.

Feedback

5. Select a short error from the list to list and display all shorts for a specific net (Figure 30).

   You can expand or highlight to analyze and debug the open and short errors in the Layout Editor window.

*Figure 30 Display Information for the Selected Short Error*



Feedback

## Analyzing Parasitics Using StarRC Virtuoso Integration

The Virtuoso Integration (VI) interface with the Cadence® Virtuoso® custom design platform includes the **StarRC** menu. From the **StarRC** menu, you can analyze parasitics using the following features:

- **Parasitic Generation Cockpit**
- **Parasitic Prober**
- **3D Viewer**
- **Opens Debugger**

This topic describes in detail how to perform tasks using the **Parasitic Prober** and **Parasitic Explorer** menus:

- Using Parasitic Prober in the Parasitic Explorer Flow

To use the Parasitic Explorer tool, you should specify the following commands during a StarRC run:

- `REDUCTION: NO`
- `NETLIST_TAIL_COMMENTS: YES`
- `EXTRA_GEOMETRY_INFO: NODE RES`

For detailed information to use the commands, see the *StarRC User Guide and Command Reference*.

Feedback

## Using Parasitic Prober in the Parasitic Explorer Flow

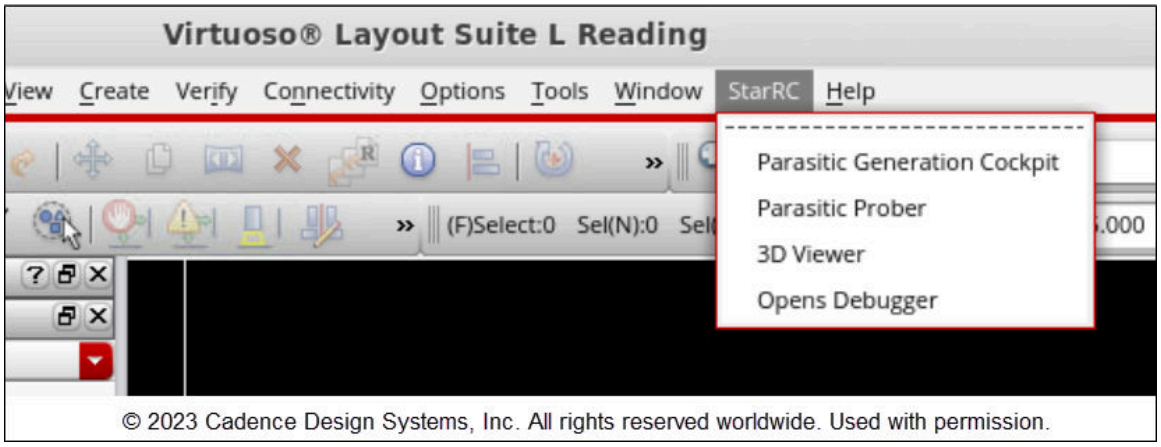
The Virtuoso Integration (VI) interface with the Cadence® Virtuoso® custom design platform allows to use **Parasitic Prober** in Parasitic Explorer mode and perform the following tasks:

- Specify the GPD path and load the GPD or the SPF file.
- Run in **PE Mode** after loading the GPD or the SPF file.

To launch the Parasitic Prober GUI from the Virtuoso Integration menu bar and to perform probing,

1. Start the GUI from the StarRC OA View or Layout View.
2. Choose **StarRC > Parasitic Prober** from the Virtuoso Integration menu bar (Figure 31).

*Figure 31 StarRC Menu in Virtuoso*



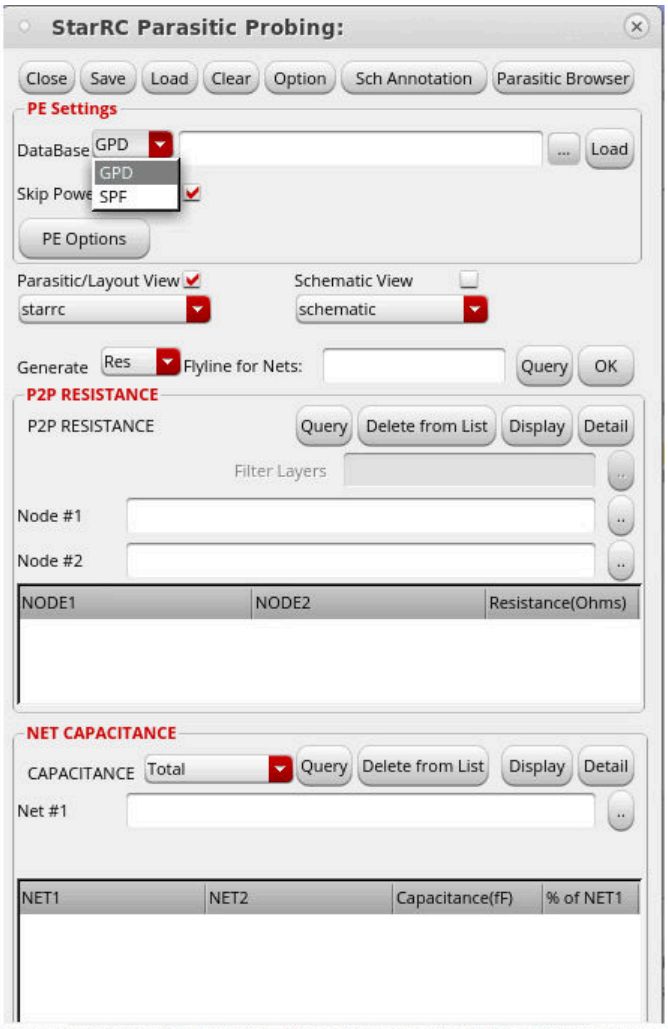

The Virtuoso Integration tool displays StarRC Parasitic Probing window (Figure 32).

3. Choose either **GPD** or **SPF** from the **DataBase** drop-down menu to select either the GPD or a SPF file and load the file.

   If the **Skip Power Nets** check box is selected, the Parasitic Explorer tool does not load the power nets before loading the GPD or the SPF file.

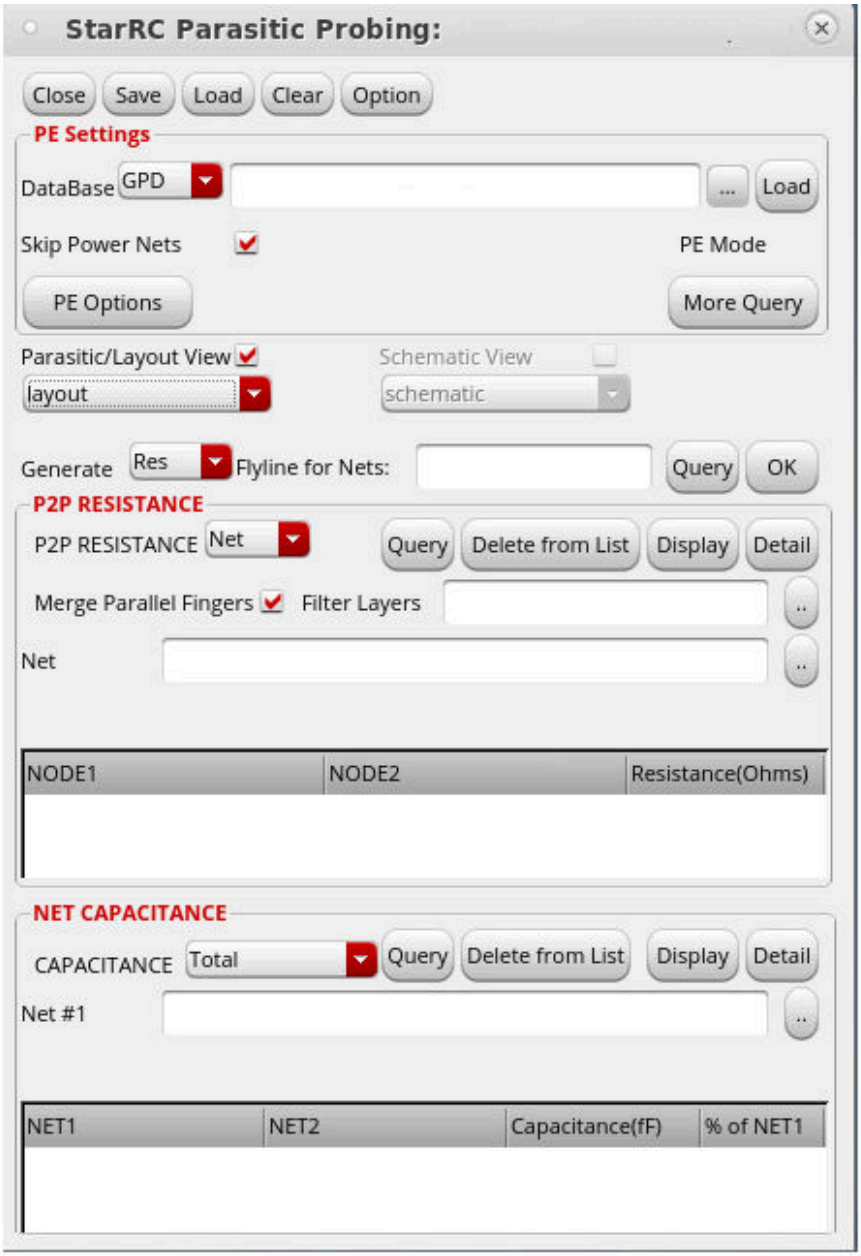
Feedback

*Figure 32 StarRC Parasitic Probing*



Feedback

4. Select **layout** from the **Parasitic/Layout View** drop-down menu to view annotations in the layout view after loading the GPD or the SPF file.



### Loading a GPD or a SPF File

To load a GPD,

**Note:**

You can perform the following steps to load a SPF file too.

1. Specify the GPD path after selecting **GPD** from the **DataBase** (Figure 32) drop-down menu.

   The tool automatically populates the **GPD** path.

2. Click **Load** to load the specified GPD file.

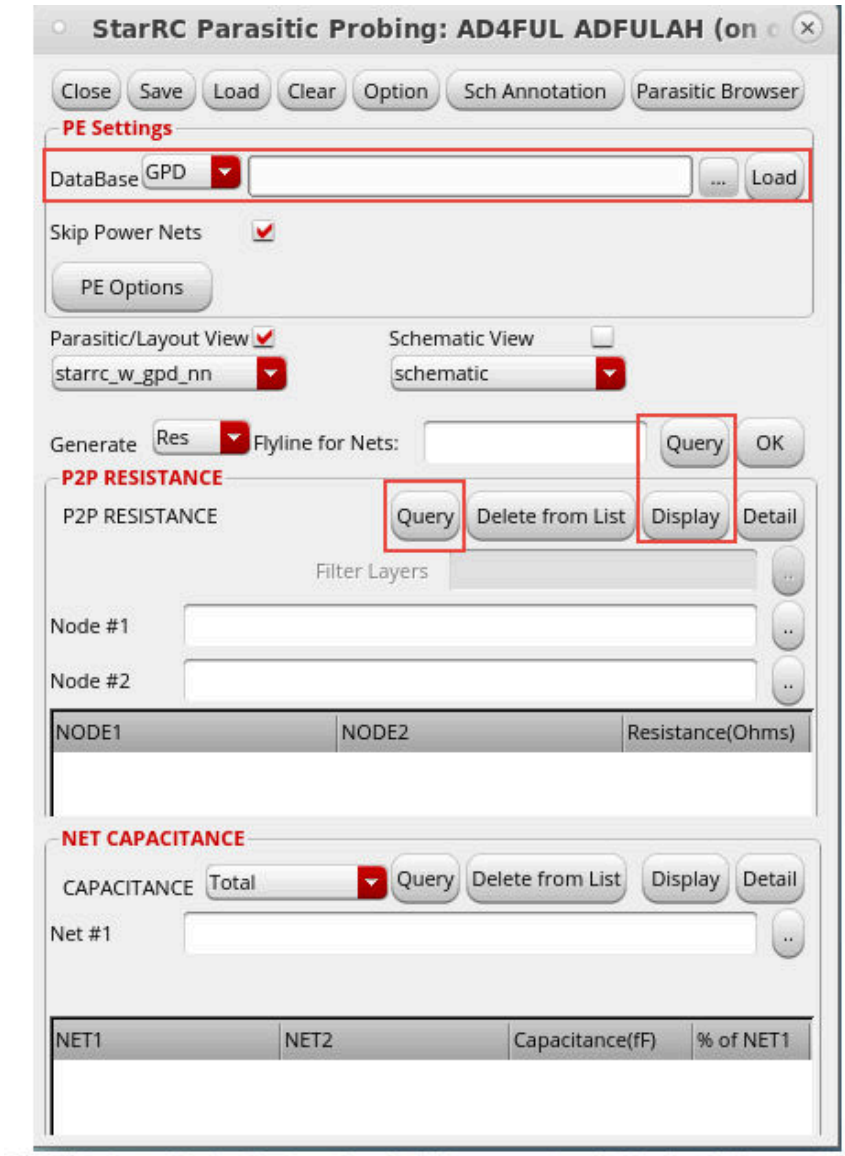
   However, if the path to the GPD does not exist, the tool issues the warning message and allows you to browse and select the correct path for the GPD directory.

   To browse and select any file or folder, click  (near **Load**) to browse through directories and select the required GPD file.

   The StarRC shell is launched in the background if the GPD loads correctly.

Feedback

*Figure 33 Loading GPD*

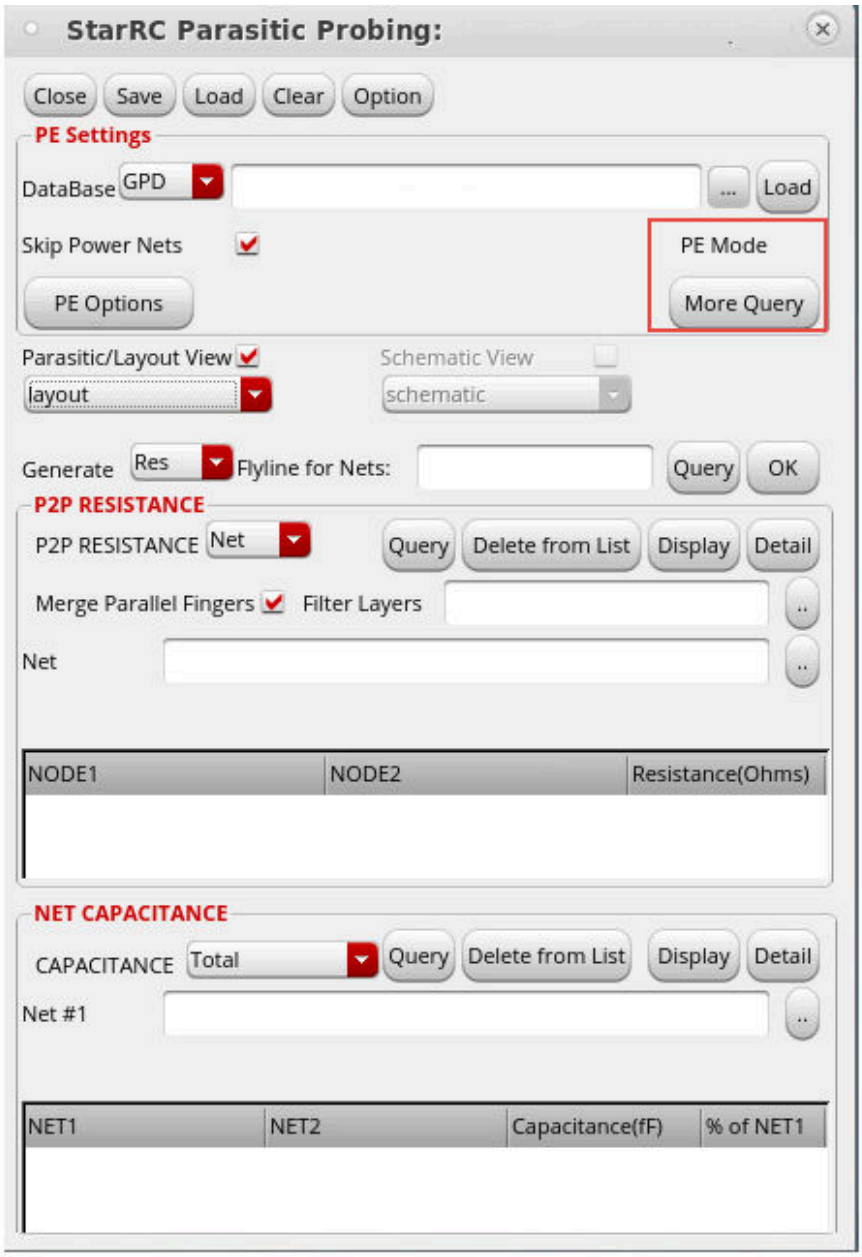





After loading the GPD, **PE Mode** appears in the StarRC Parasitic Probing window (Figure 34).

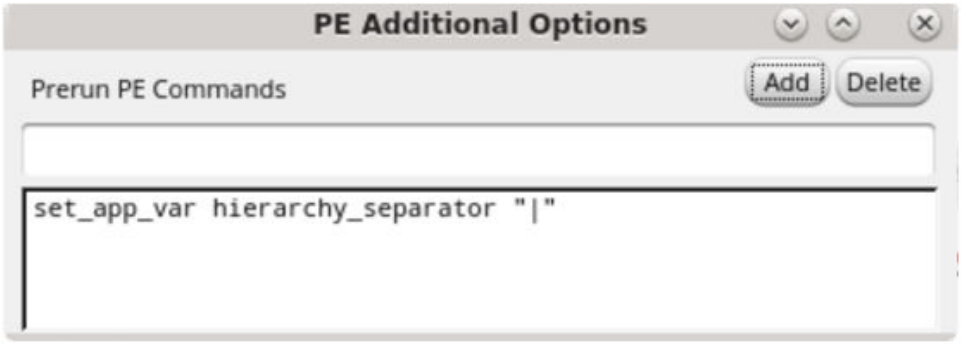
Feedback

*Figure 34 PE Mode with More Query option is visible*






Before loading the GPD or the SPF file, you can also set any commands as needed in the StarRC shell in the PE Additional Options window. To do this, click **PE Options** to open the PE Additional Options window. The specified commands are run in the StarRC shell.






3. Click **More Query** under **PE Mode** to open the More Parasitic Explorer Functions window.

   For more information, see Accessing Additional Parasitic Explorer Functions in Probing.

### Querying Resistance

To query resistance, choose either **Net** or **Node** from the **P2P RESISTANCE** drop-down menu. The default is **Net**. Select the node or net name either by performing one of the following methods:

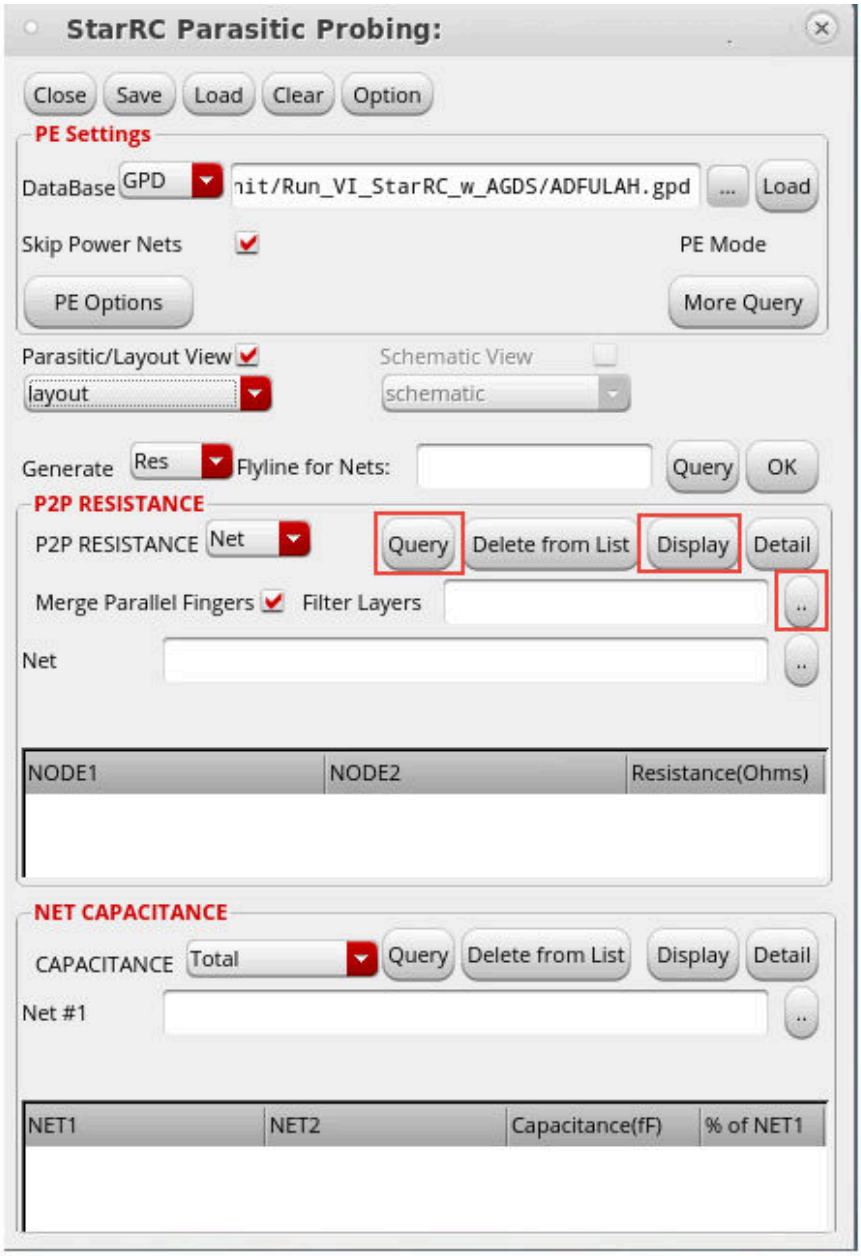
- Selecting from the extracted view by clicking **Query**.
- Selecting from the net browser by clicking  .
- Typing the name of a net in the Node fields or the Net field.

**Note:**

If **PE Mode** does not appear after loading the GPD or the SPF file, the **P2P RESISTANCE** drop-down menu is not available.

Feedback

*Figure 35 Selecting Net or Node to view point-to-point resistance*






In Figure 35, for example, to see additional information about NODE1 and NODE2 in the **Parasitic Netlist Browser NODE** window,

- Click B1 under NODE1 (Figure 36)
- Click XI1| X14|M1|G under NODE2 (Figure 37)

Feedback

*Figure 36 Displaying Additional Information of NODE 1*

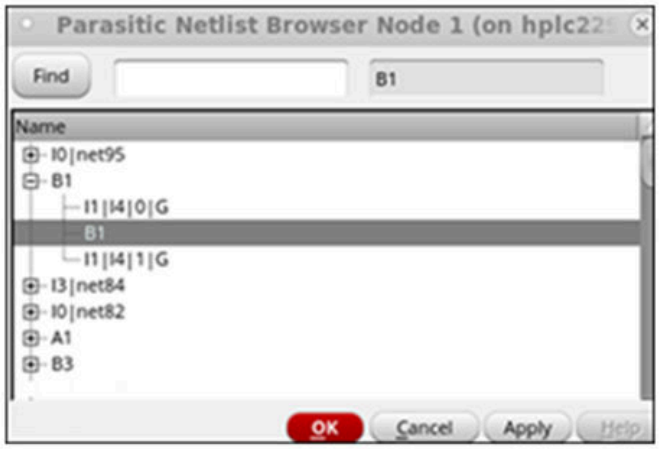
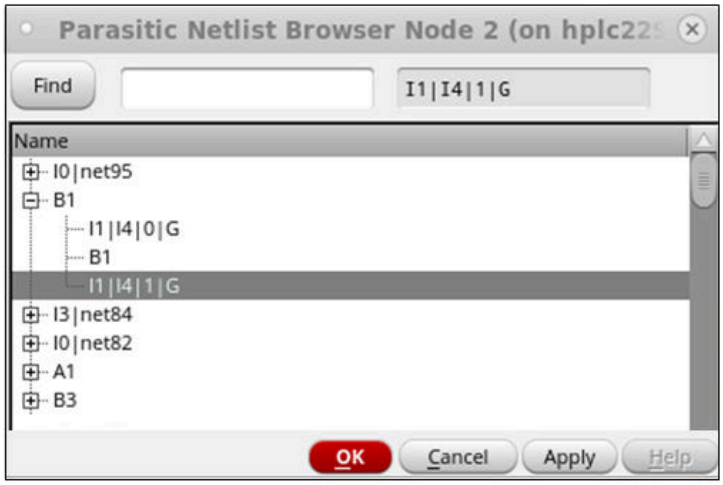




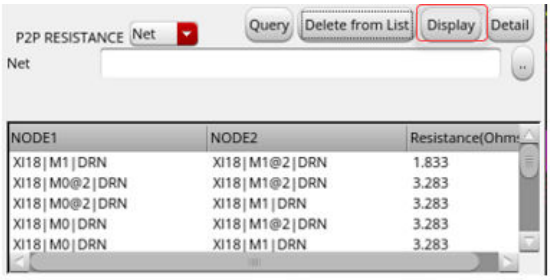

*Figure 37 Displaying Additional Information of NODE 2*






After specifying node pairs, click **Display** for the tool to list point-to-point resistance between the nodes (Figure 35).

After specifying the Net name, click **Display** for the tool to list point-to-point resistance between different pairs of nodes on the specified net.





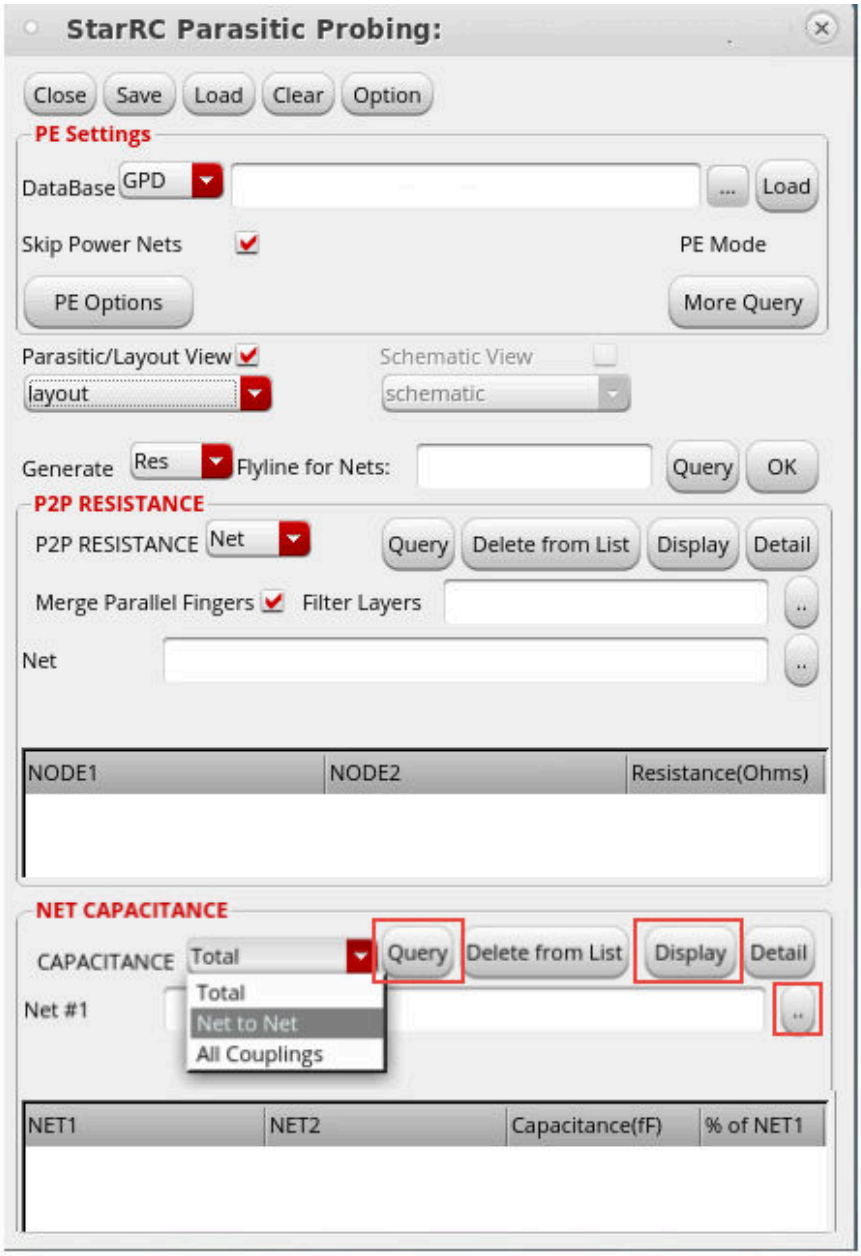

### Querying Capacitance

To query capacitance, choose either **Total**, **Net to Net**, or **All Couplings** from the **Capacitance** drop-down menu. The default is **Total**, which is total capacitance. Then, select the node or net name either by performing one of the following methods:

- Selecting from the extracted view by clicking **Query**.
- Selecting from the net browser by clicking 
- Typing the name of a net in the Net field.

Feedback

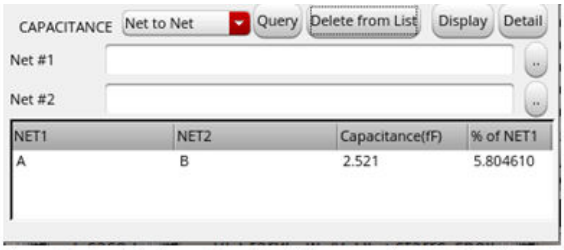
*Figure 38 Total Capacitance Query*





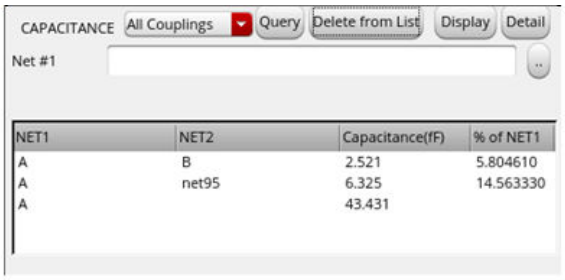

For **Total**, the total capacitance for the specified net is displayed (Figure 38).

For **Net to Net**, the net-to-net capacitance lists coupling capacitance of **NET1** and **NET2**.






For **All Couplings**, the all coupling capacitances on the specified net are displayed.



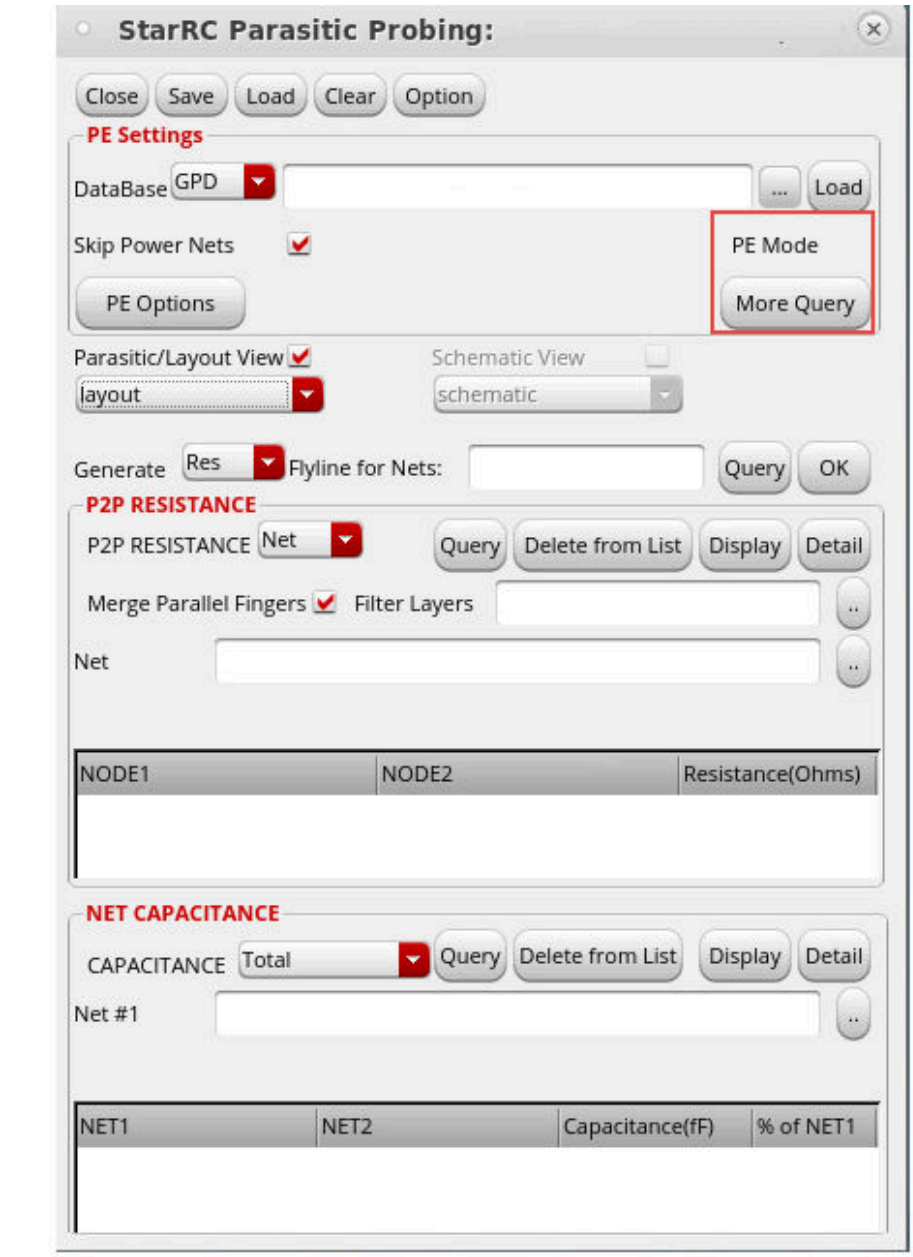



### Accessing Additional Parasitic Explorer Functions in Probing

To see additional parasitic functions, click **More Query** (Figure 39) under **PE Mode** to open the More Parasitic Explorer Functions window (Figure 40).

Feedback

*Figure 39 PE Mode with More Query option is visible*

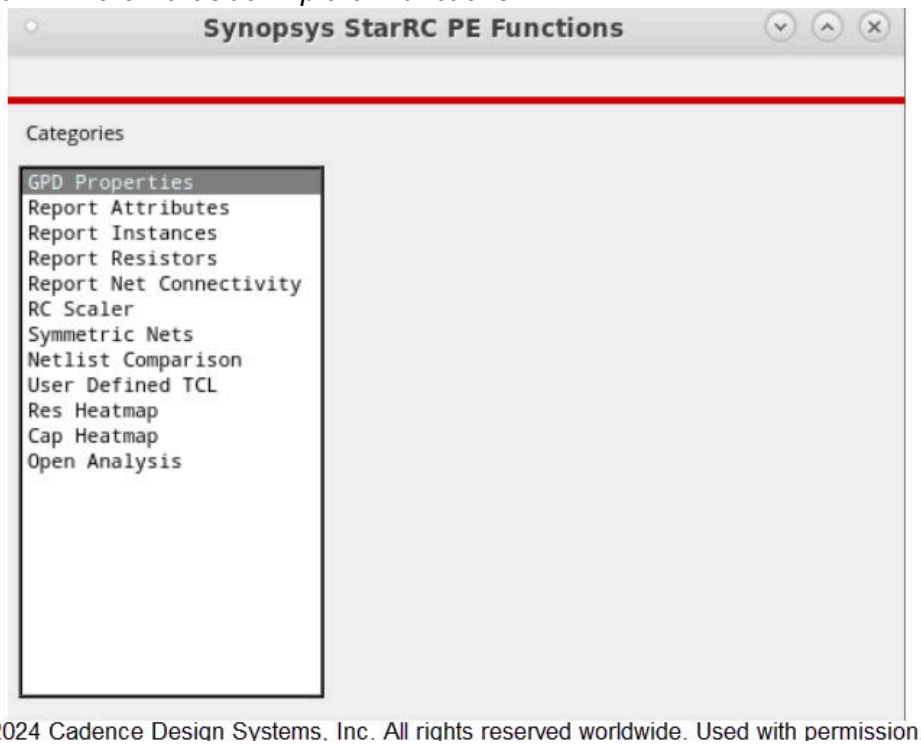





To know when **PE Mode** appears in the StarRC Parasitic Probing window, see Loading a GPD or a SPF File.

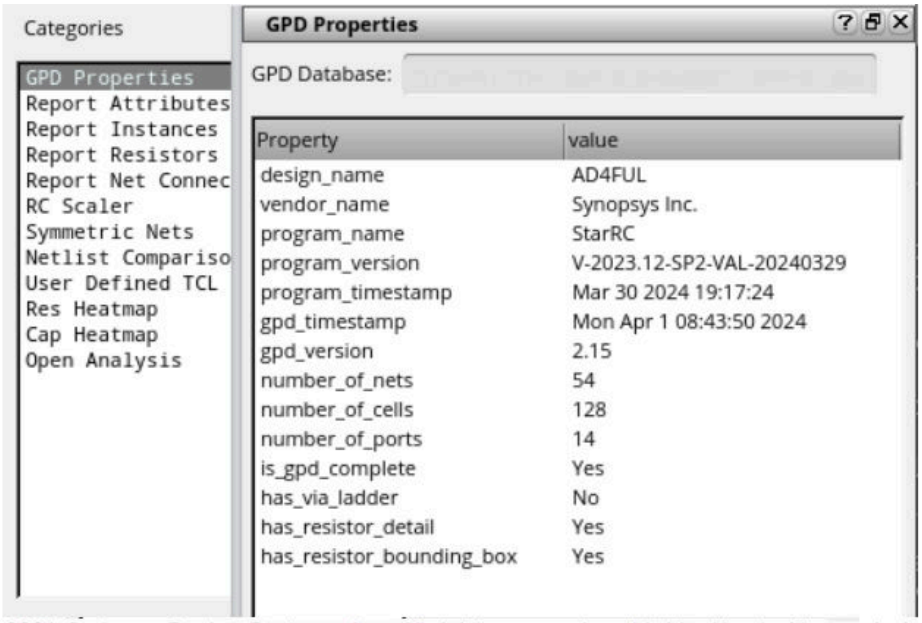
Feedback

*Figure 40 More Parasitic Explorer Functions*



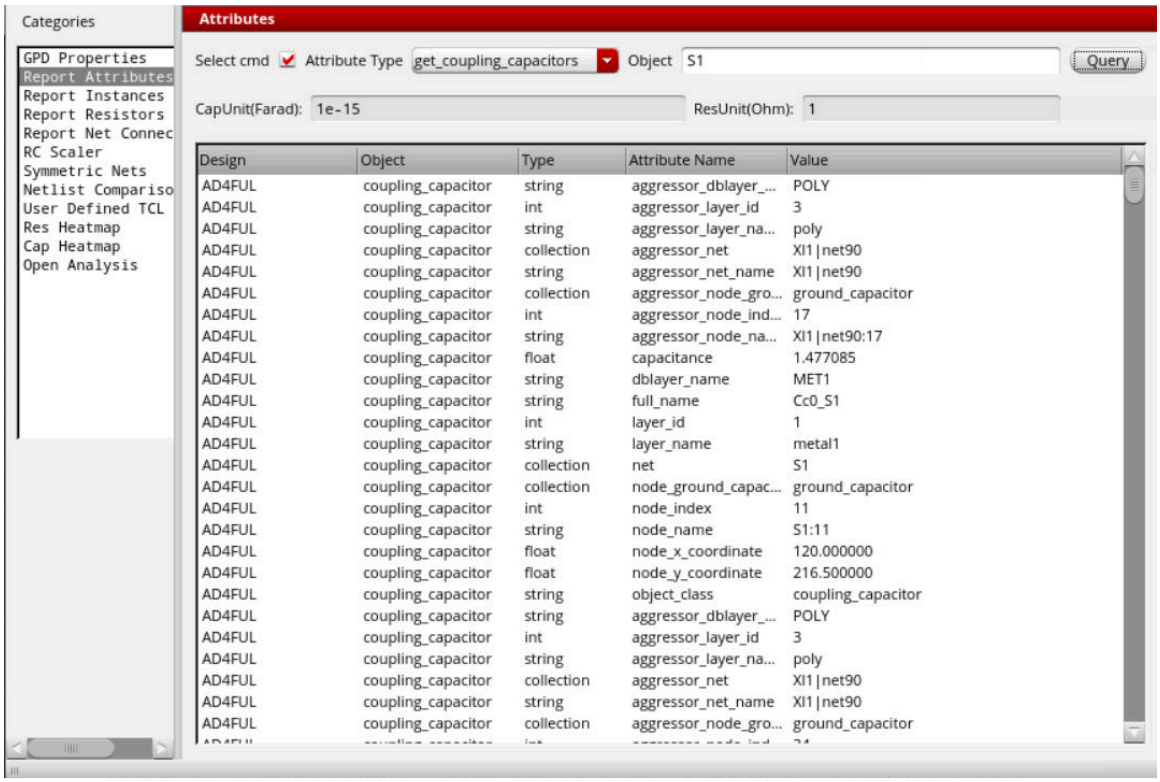
From the More Parasitic Explorer Functions (Figure 40) window, you can perform any of the following tasks:

- Click **GPD Properties** and specify a GPD directory in the **GPD Database** box, and click **Query** to view the properties of the uploaded GPD directory.



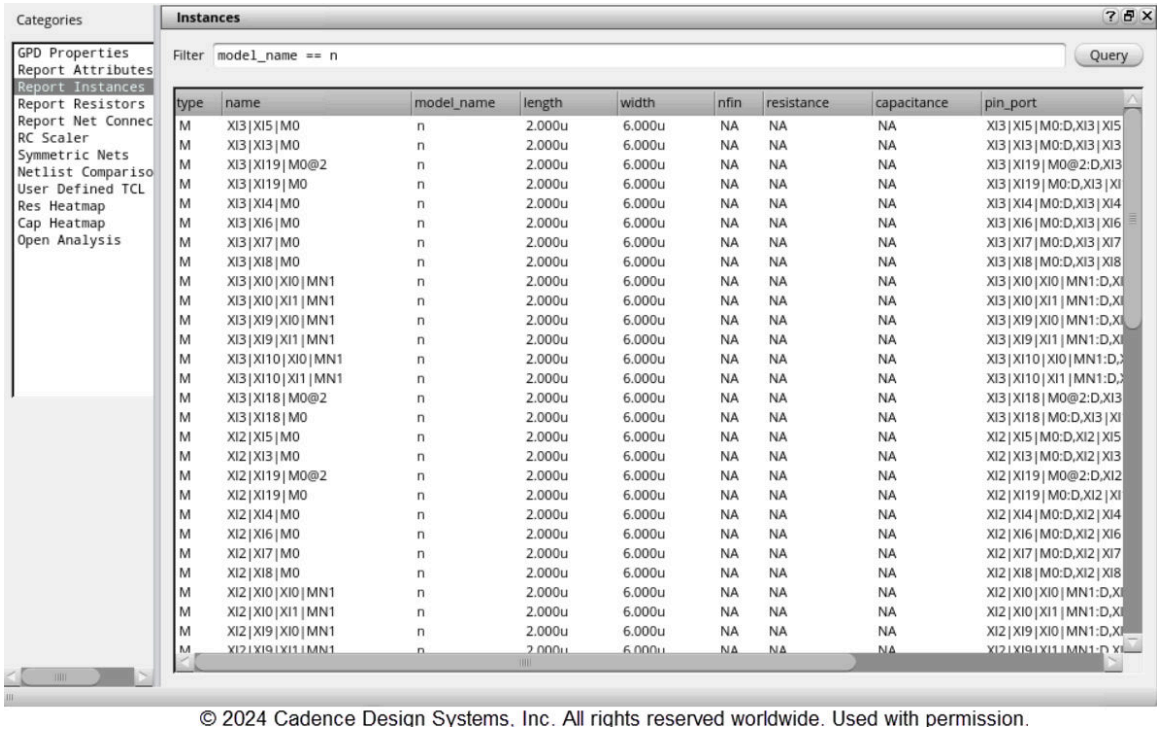


- Click **Report Attributes** and choose the Tcl command from the **Attributes** drop-drop, and click **Query** to display the report.



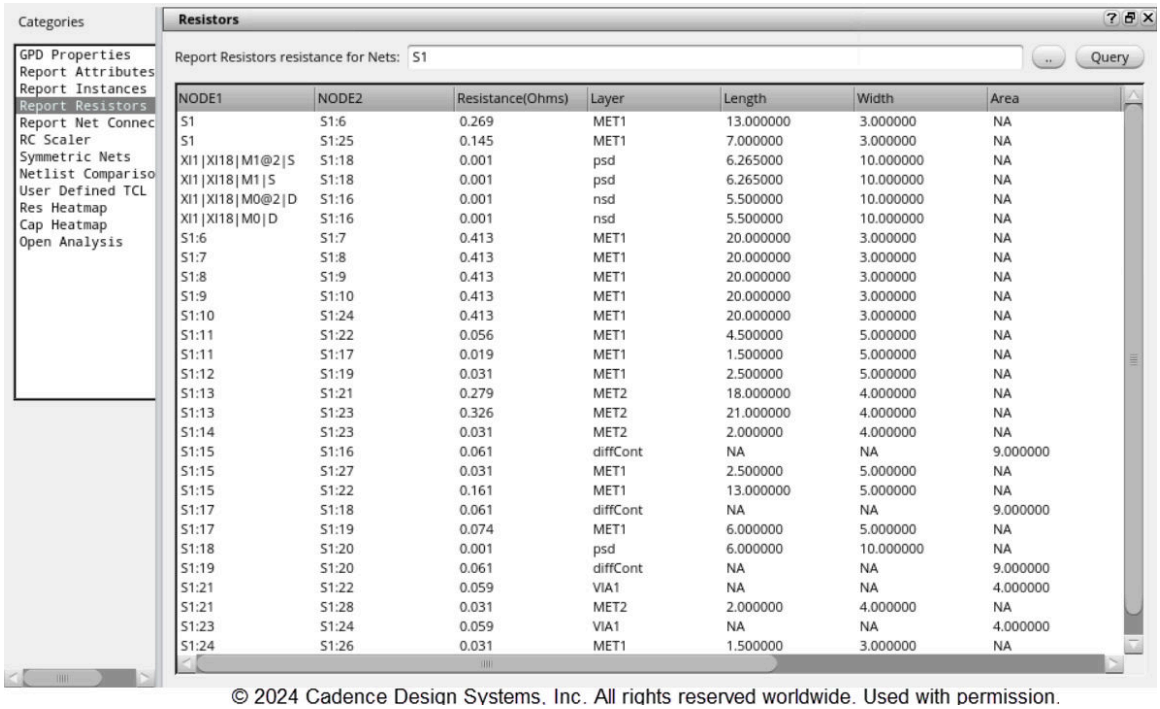



- Click **Report Instances** and specify a cell name in the **Filter** box, and click **Query** to display details of instances based on the specified filter.



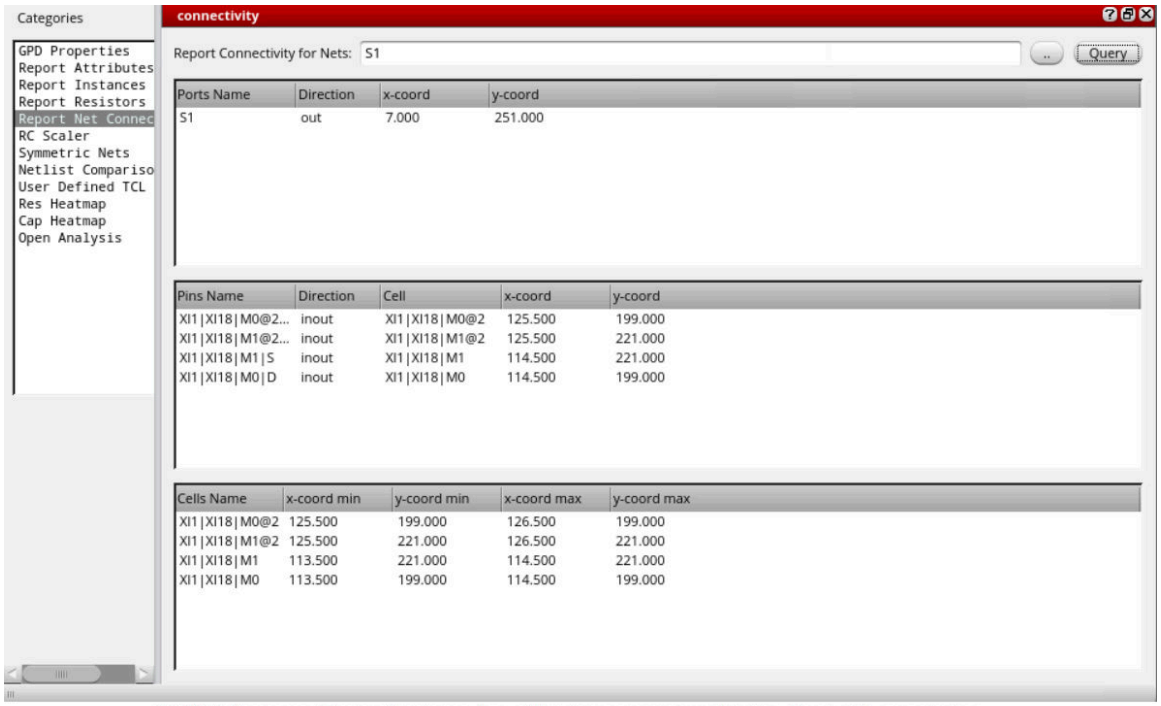



- Click **Report Resistors** and specify a net name in the **Report Resistors resistance for Nets** box, and click **Query** to report resistance of the specified net.





- Click **Report Net Connectivity** and specify a net name in the **Report Connectivity for Nets** box, and click **Query** to report names of ports, pins, and cells of the specified net with their direction and x and y-coordinates.





- Click **RC Scaler** to open the RC Scaler window and generate net-based or instance-based reports.

  1. **RC Scaler Config File**: Specify the configuration file or load the configuration file. You can also edit and save the file if needed.
  2. **Scaled GPD Name**: Specify or load the output scaled GPD file that can be subsequently converted into an SPF or SPEF file.

     When you click **Apply**, the `scale_parasitics` command runs in the StarRC tool. For more information, see scale_parasitics.

  - **Generating the Net-based Report:**

    To generate the net-based report, specify the following details:

    1. **Mode**: Select either **Net** or **Node** mode. The default is **Net**.

       If you select **Node** mode, then you can specify the start point and end point of a net in the **-from** and **-to** fields.

    2. **net_list**: Specify or load the names of nets. You can specify multiple net names.
    3. **layers**: Specify or load the database layer name for mapping design layers. You can specify multiple layer names.

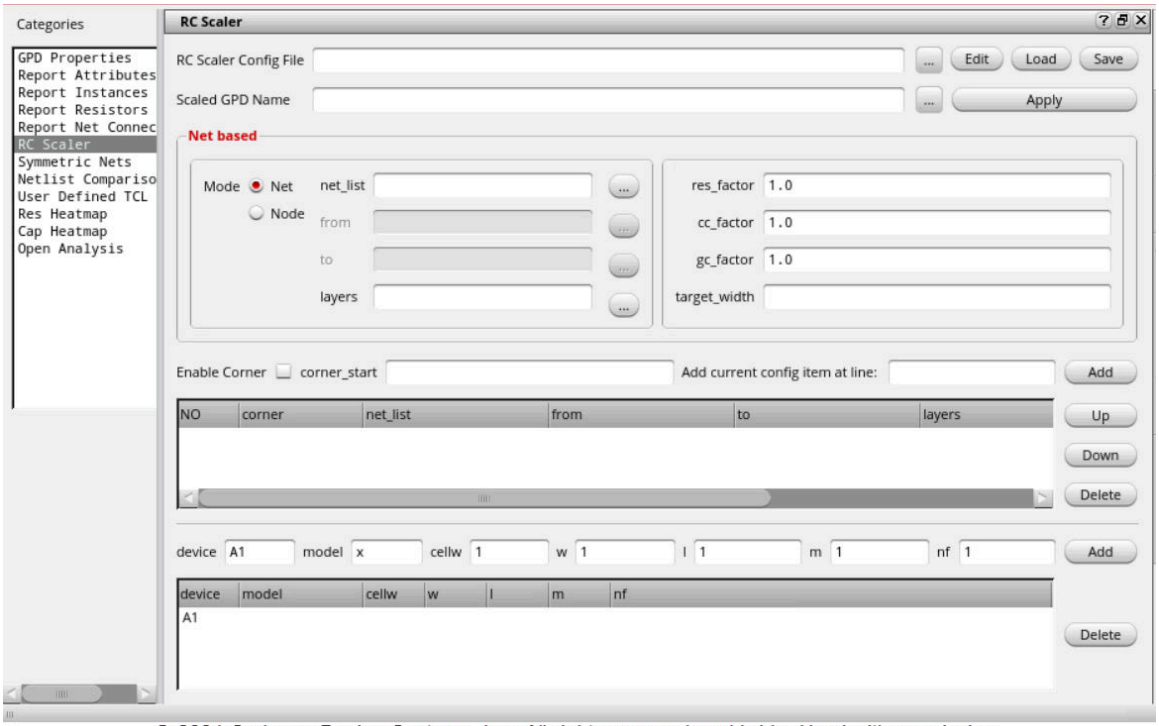
4. **res_factor**: Specify the scaling factor for resistance. By default, the tool sets the scale factor to 1.0.
5. **cc_factor**: Specify the scaling factor for coupling capacitance. By default, the tool sets the scale factor to 1.0.
6. **gc_factor**: Specify the scaling factor for ground capacitance. By default, the tool sets the scale factor to 1.0.
7. **target_width**: Specify the resistor width to scale the resistors.
8. (Optional) **Enable Corner**: Select the **Enable Corner** check box to add a corner name for the specified corner.
9. **Add current config item at line**: Specify the line number in which the configuration file needs to be added and then click **Add**.

Feedback

Note the requirements to specify the line number:

- The value must be an integer.
- If the value is not an integer, the value less than 1, or the value is greater than the netlists available, then the netlist specified is added as the last netlist.
- The value must not be greater than the number of items available in the configuration file.

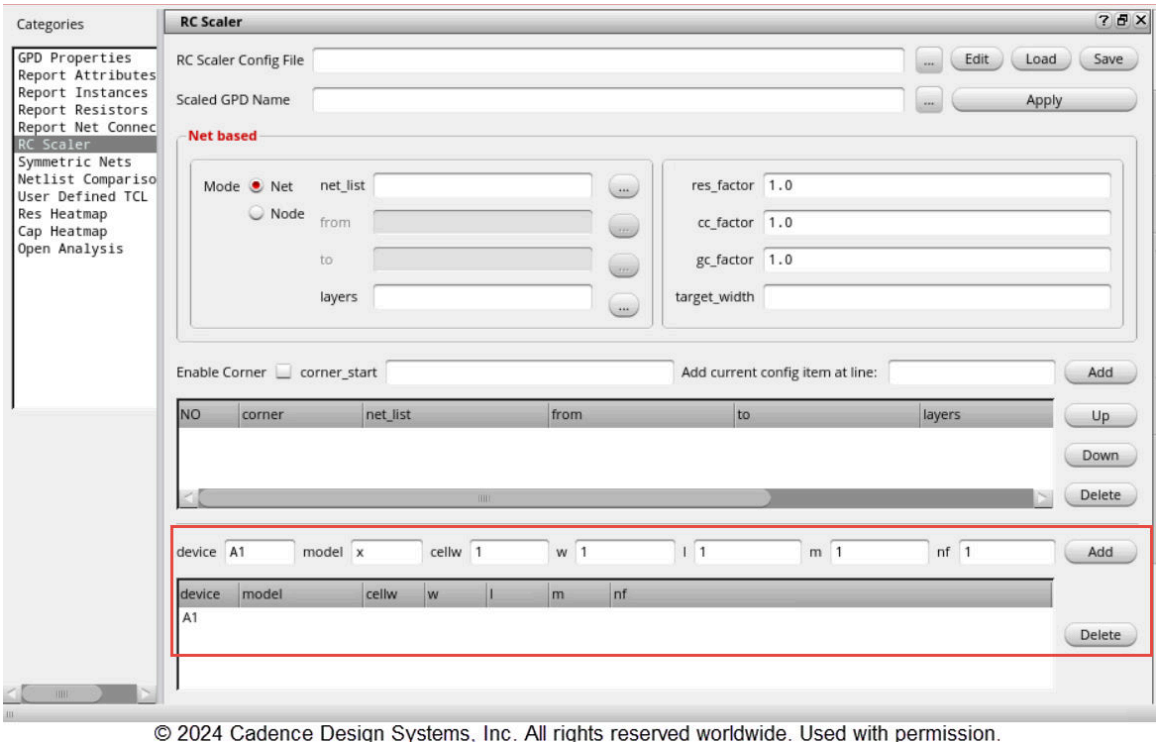
10. Click **Up** or **Down** to adjust the position or order of the current selected item.



Feedback

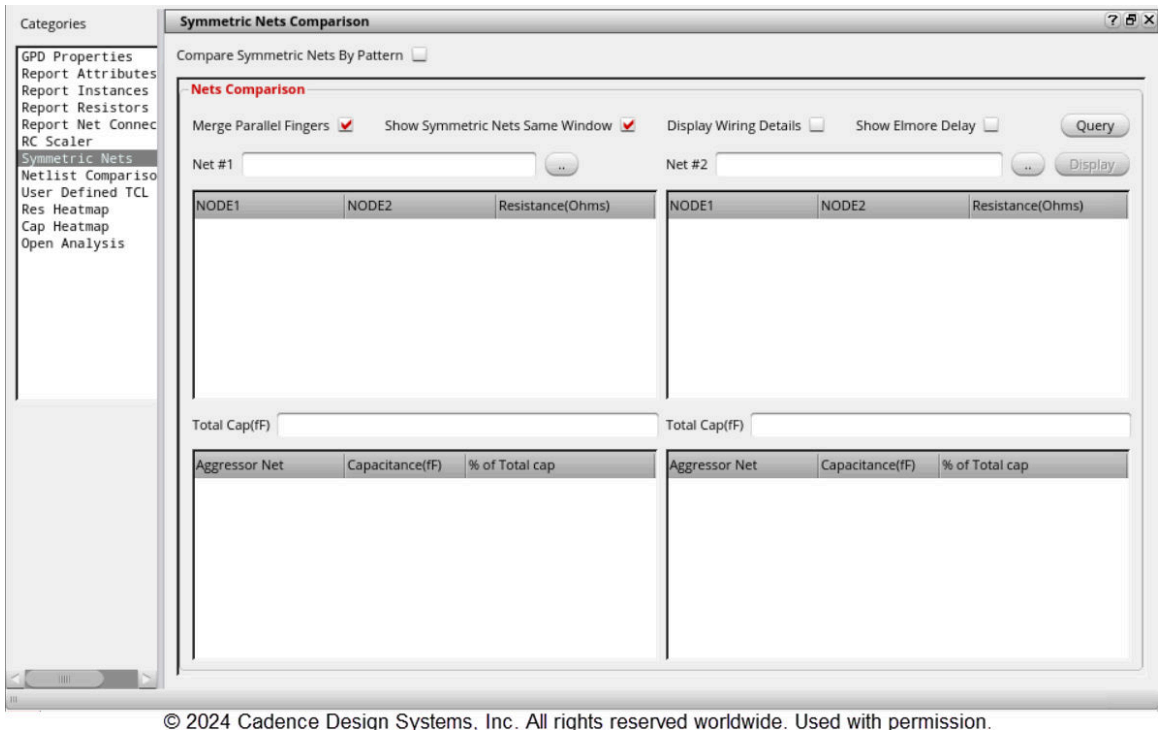
- **Generating the Instance-based Report**

  To generate the instance-based report, specify the device instance name in the **device** field. Enter all other fields if needed. For more information regarding the fields, see scale_parasitics.



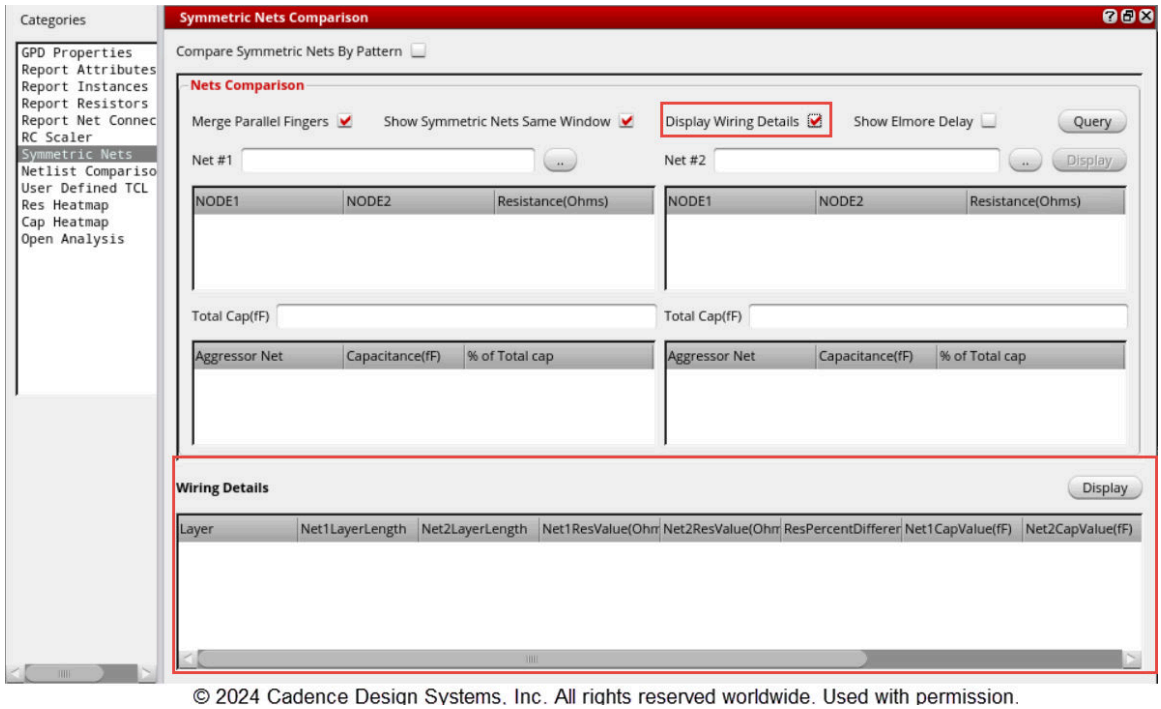



- Click **Symmetric Nets**
  - Viewing **Nets Comparison** report
    - Specify net pairs in the **Net #1** and **Net #2** field respectively, and click **Query** to compare the specified nets and check whether they are symmetrical. Also, you can
    - Click **Display** to view side-by-side report and the Net Comparison report
    - Select a layer or layers and click **Display** to view the selected layers in the extracted view






- To see the writing details in the **Writing Details** box, check the **Display Writing Details** check box.
- Select a layer or layers and click **Display** to view the writing details of the layers.

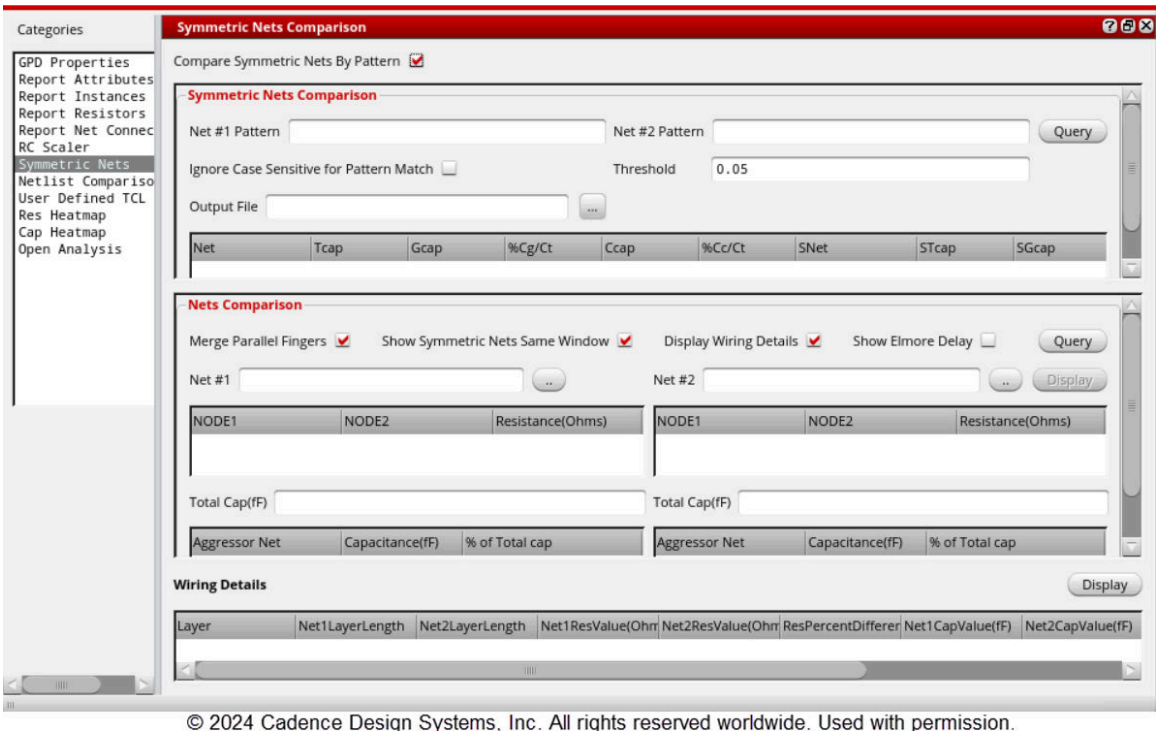





- Viewing **Symmetric Nets Comparison** report

Feedback

1. Select **Compare Symmetric Nets By Pattern** to display **Symmetric Nets Comparison** options.

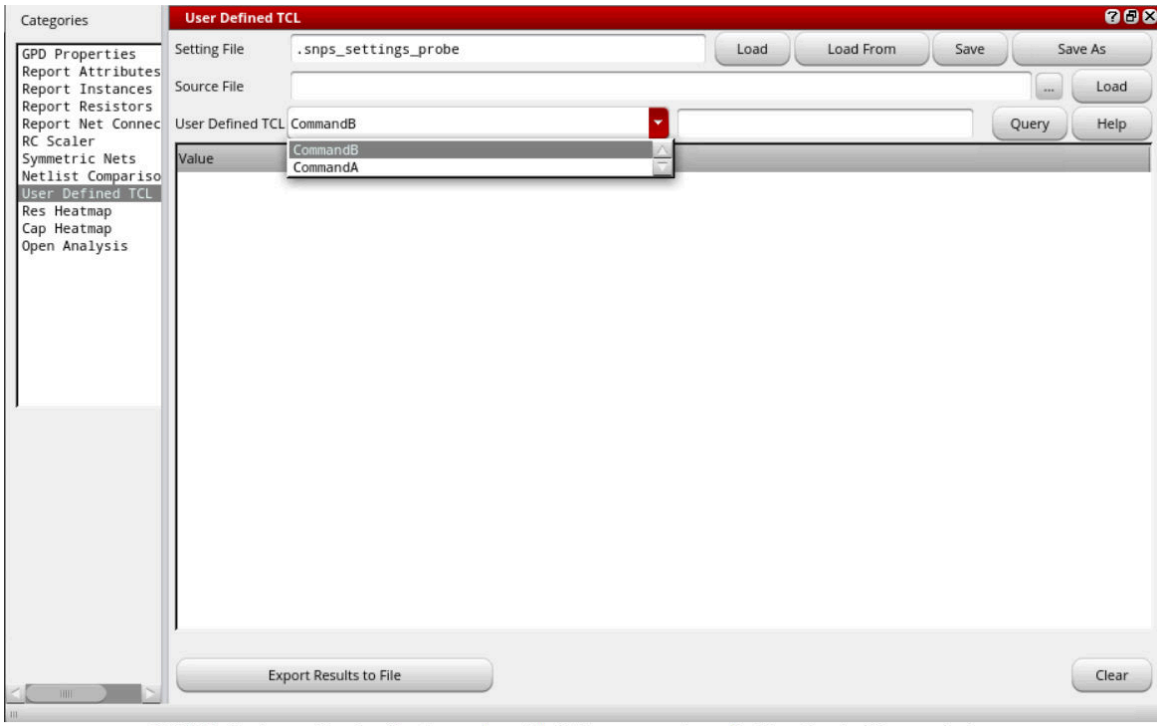





2. Specify net patterns in the **Net #1 Pattern** and **Net #2 Pattern** field respectively, and click **Query** to compare capacitance of the specified symmetric nets.

   For information about using the options in **Symmetric Nets Comparison**, see the report_compare_symmetric_nets_capacitance.

- Click **User Defined TCL** and specify the required files in the following fields.






*Example 8 Specifying Setting in .snps_settings_probe File*

```
# Settings to specify in the .snps_settings_probe file to use in the
 User Defined TCL GUI

PE_TCL_FILE: lock_name/label_name/design/database/example.tcl
PE_TCL_COMMAND: CommandA CommandB
```

1. **Settings File**: Specify the `.snps_settings_probe` file

   Or

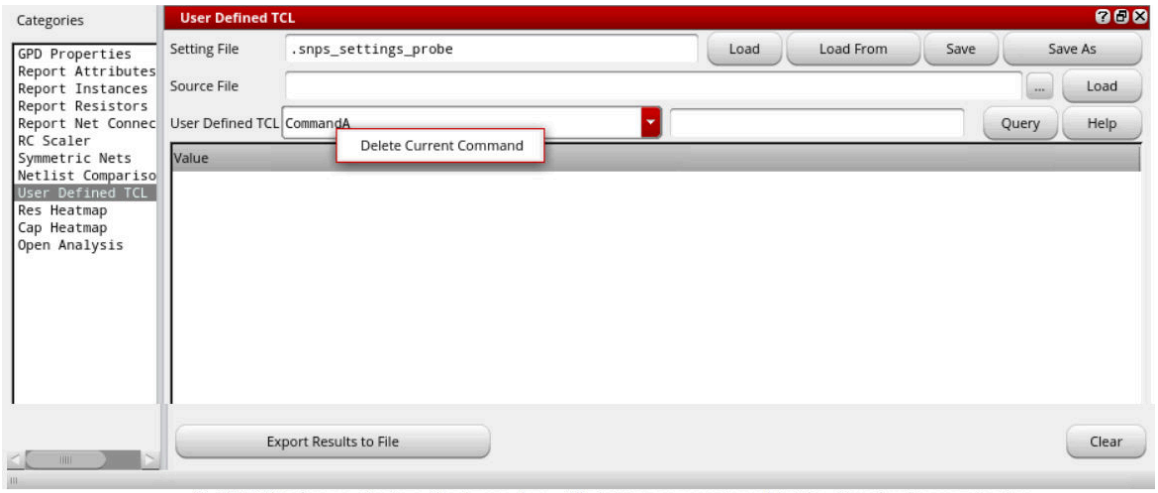
   **Source File**: Specify the Tcl file that should be sourced and click **Load** to load the file.

   For example, you can create a PE_TCL_FILE.tcl file and add the path in the `.snps_settings_probe` file.

2. **User Defined TCL**: Choose the command from the drop-down and click **Query**.

   For example, the commands specified with the PE_TCL_COMMAND (see Example 8) are listed in the drop-down.

   To delete the command, right-click and select **Delete Current Command**.

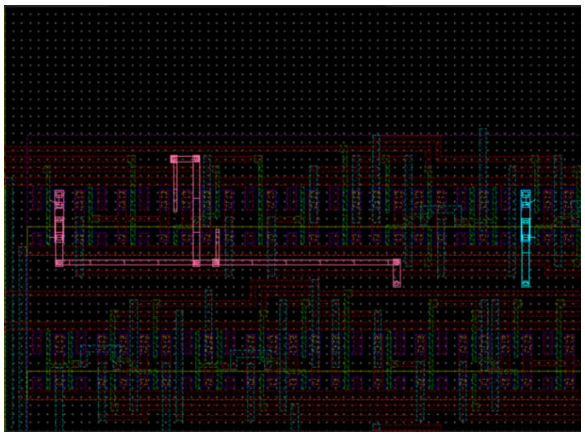




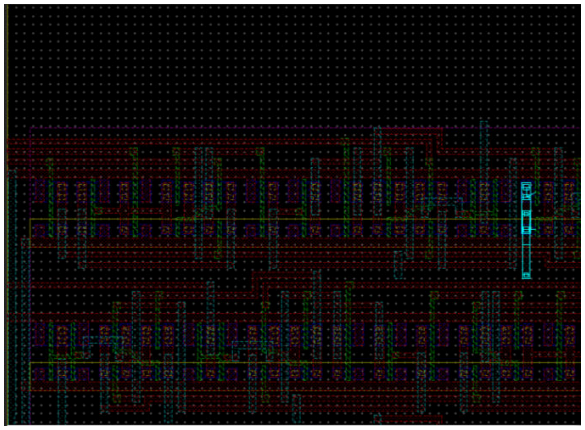

To save the results after querying, click **Export Results to File** or click **Clear** to remove all the settings specified in the **User Defined TCL** window.

- Click **Open Analysis** to analyze the resistively connected group (RCG) for the specified net and click ... to select nets from the **Parasitic Explorer Net Browser** window.

- Select groups and click **Query** to view all the groups in the extracted view as shown in the following figure:



- Select groups and click **Display** to view the selected groups in the extracted view as shown in the following figure:



## Viewing the Heatmap Report

You can calculate the p2p resistance and capacitance value to generate and display following heatmap report in the layout view:

- Resistance Heatmap
- Capacitance Heatmap

## Resistance Heatmap

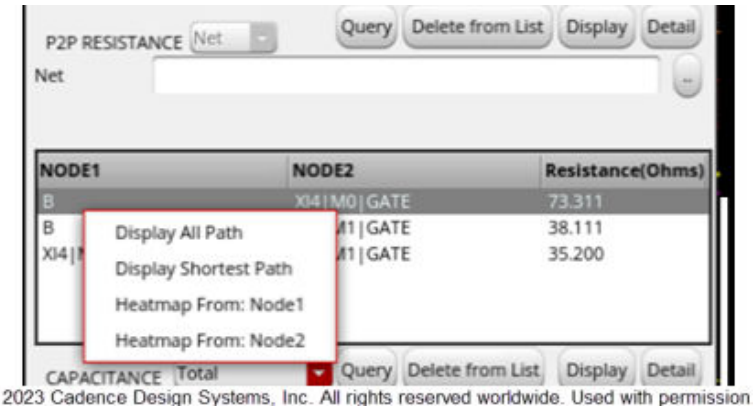
Resistance heatmap calculates the point-to-point resistance values from the start point to all the nodes on the net.

Feedback

To generate the resistance heatmap, select the heatmap with any one of the following methods:

- Right-click and select **Heatmap From: Node1** or **Heatmap From: Node2**.

  The option generates the resistance heatmap report with Node 1 or Node 2 as the start point.

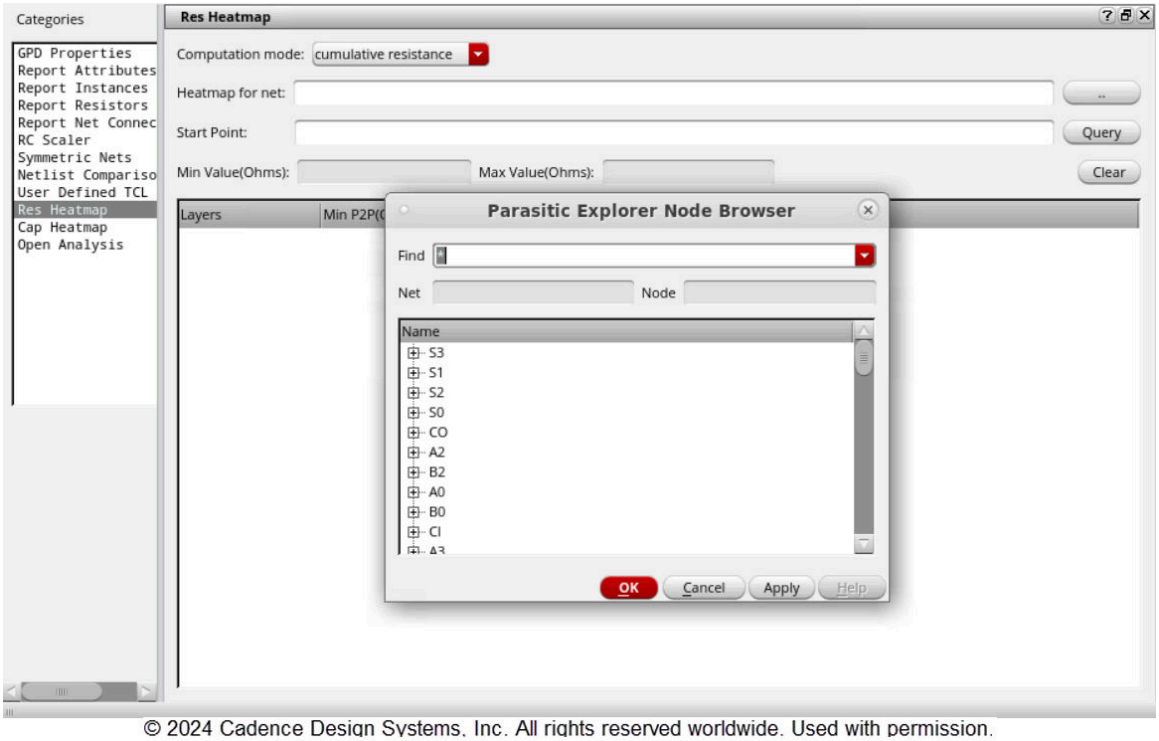




- Click **More Query** under PE Mode to open the **Res Heatmap** window.



  - Specify a net name in the **Heatmap for net** field.
  - Click  to see additional information about Node1 and Node2 in the **Parasitic Netlist Browser NODE window**.



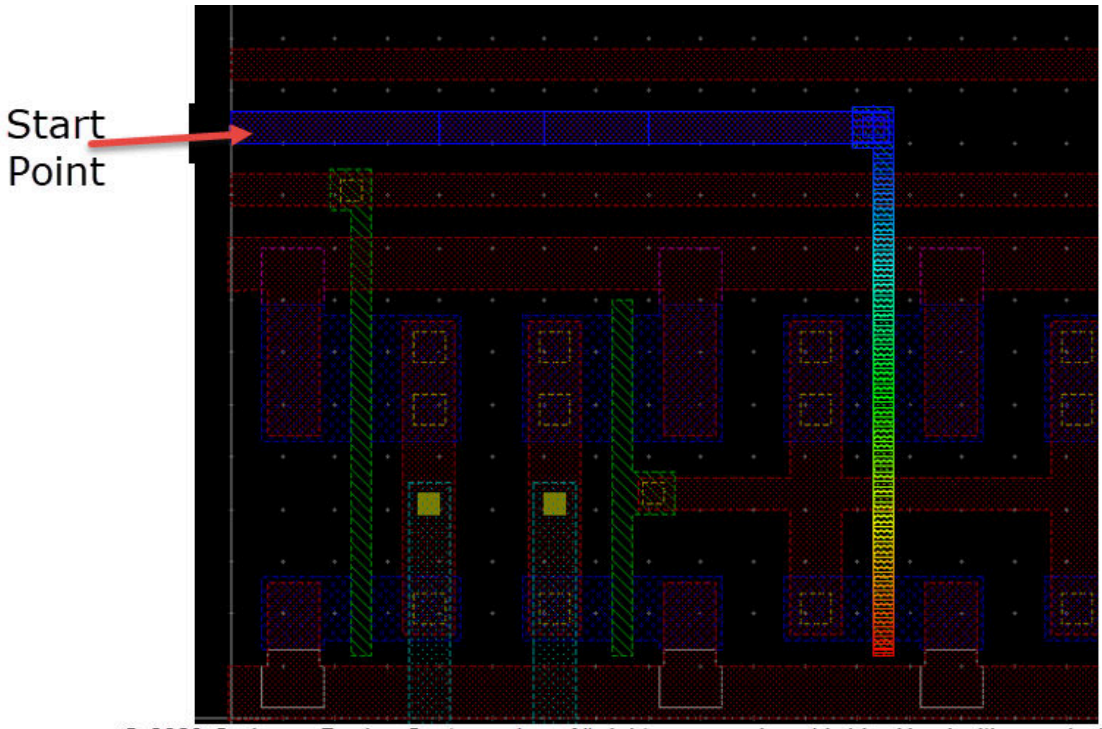



- Choose either **cumulative resistance** to display the trend of resistance change from a start point or **resistive hotspots** to display maximum resistance value on a net from the **Computation mode** drop-down menu. The default is **cumulative resistance**.
- Specify the start point for the net in the **Start Point** field or click **Query** to select the extracted view.
- Select layers and click **Display** to view the heatmap in the extracted view.
- The color in the extracted view of heatmap indicates the following (Figure 41):
  - Red indicates larger resistance values
  - Blue indicates smaller resistance values
  - Green and orange indicates in-between values

Feedback

The legend is displayed on the layout in the right side

*Figure 41 Resistance Heatmap in the Extracted View*





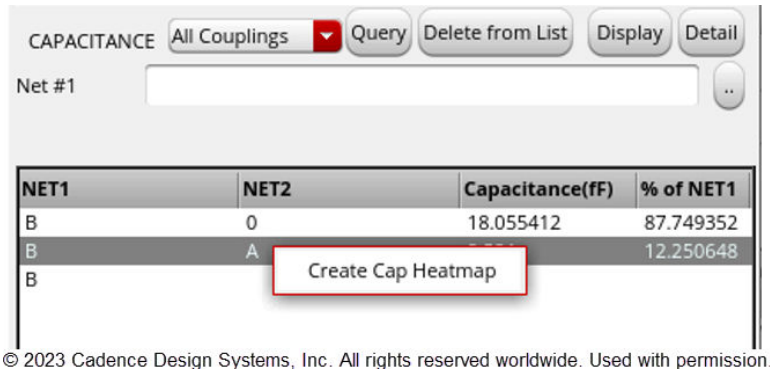

## Capacitance Heatmap

You can generate following types of capacitance heatmaps:

- **Coupling**: Calculates and displays coupling capacitance values for the nodes on the aggressor net.
- **Ground**: Calculates and displays ground capacitance values for the nodes on the target net.
- **Total**: Calculates and displays total capacitance values for the nodes on the target net.

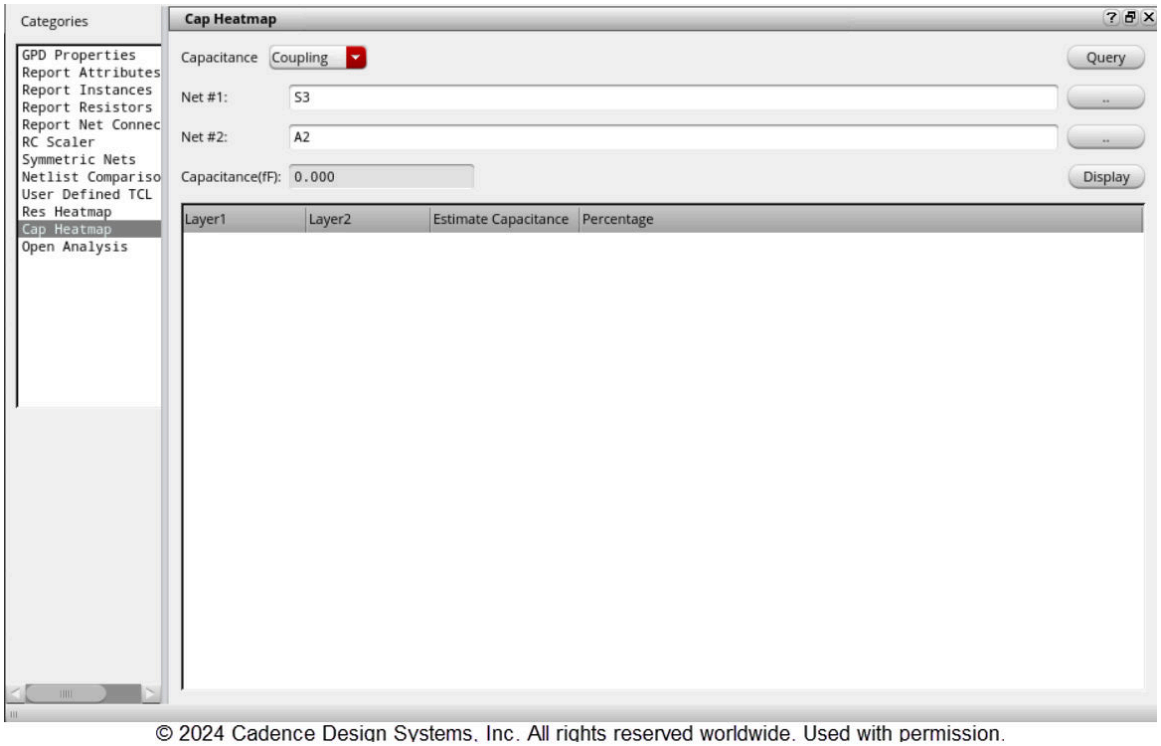
To generate the capacitance heatmap, select the heatmap of the node using any one of the following methods:

- Right-click and select **Create Cap Heatmap** for the net and aggressor net.



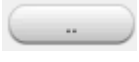


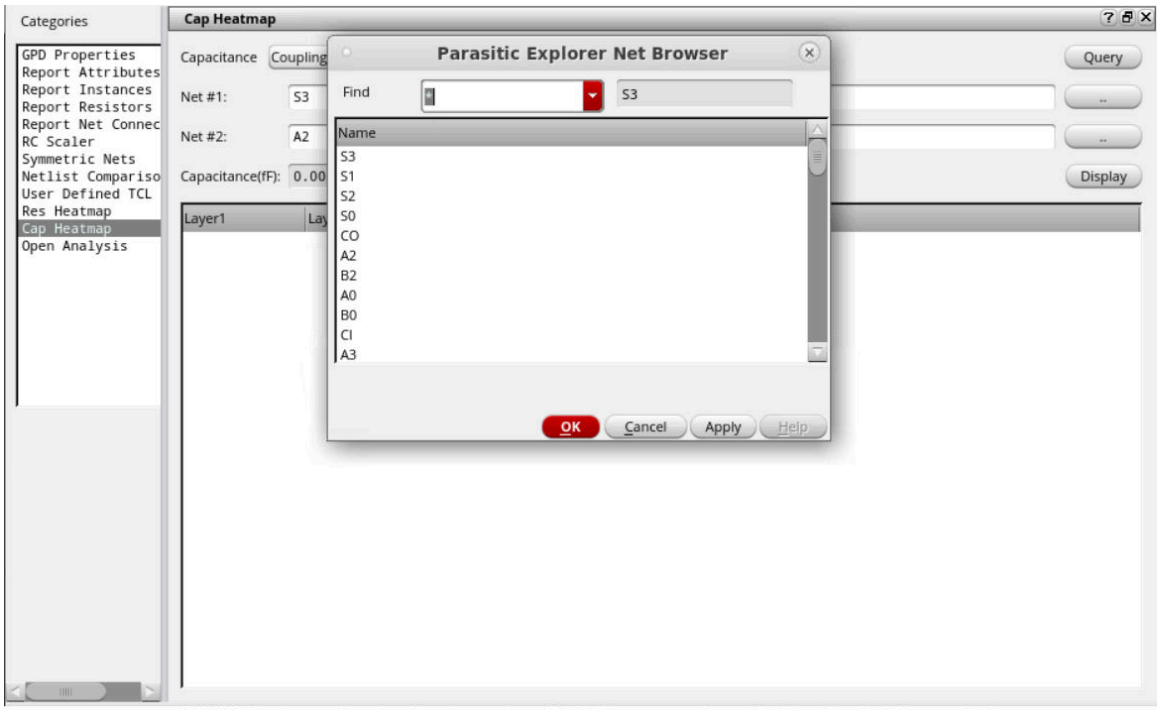

The **Cap Heatmap** window appears:






1. Choose either **Coupling**, **Ground**, or **Total** from the **Capacitance** drop-down menu. The default is **Coupling**.

2. Specify a net name in the **Net #1** and **Net #2** field or click  to select nets from the **Parasitic Explorer Net Browser** window.

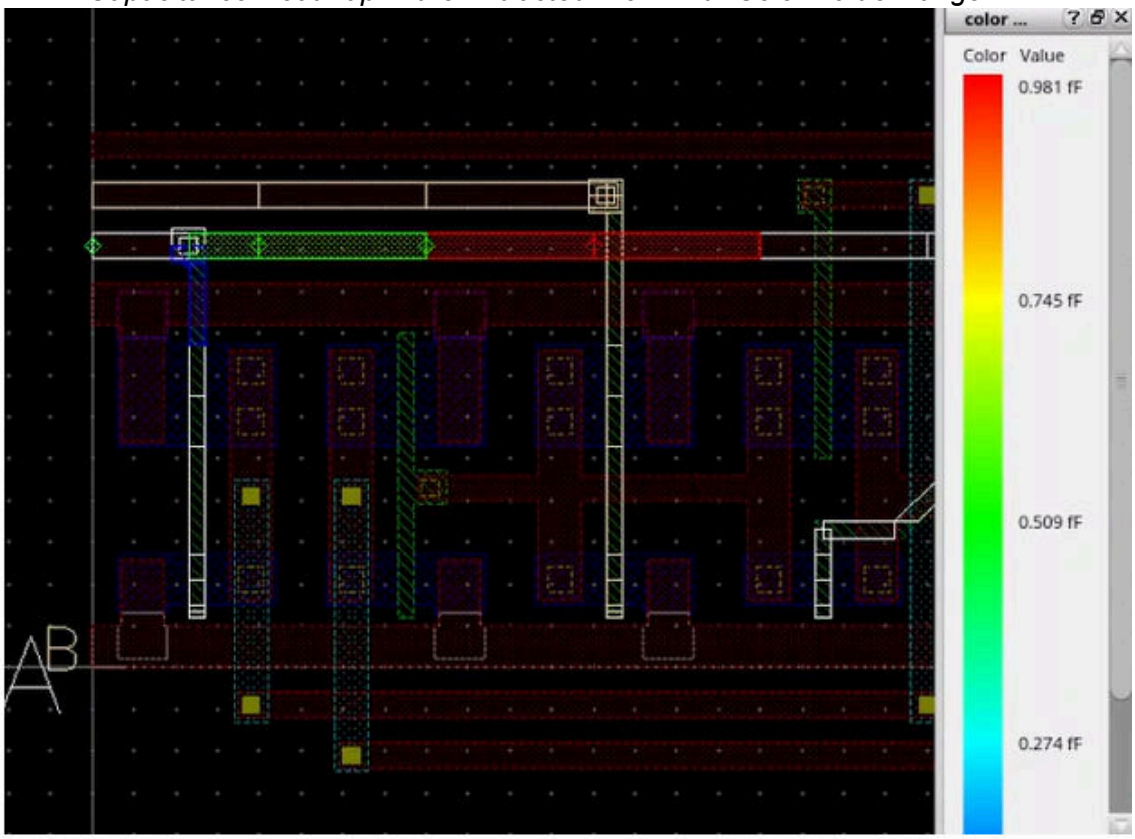





3. Select layers and click **Query** to view the heatmap in the extracted view with the color value range.

   Figure 43 displays the capacitance heatmap between net B (target net) and net A (aggressor net) with the color value range.

*Figure 42 Capacitance Heatmap in the Extracted View With Color Value Range*



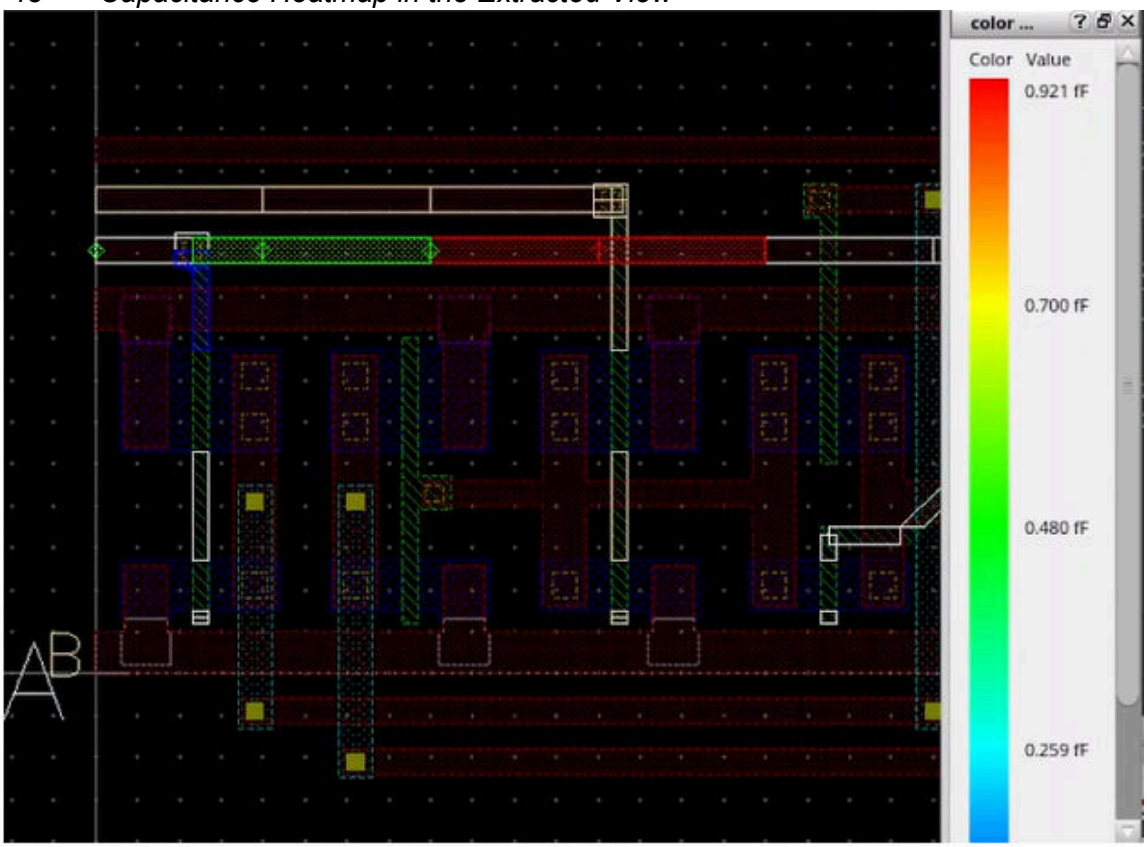



4. Select layers and click **Display** to view the heatmap in the extracted view.

   Figure 43 displays the capacitance heatmap between net B (target net) and net A (aggressor net).

Feedback

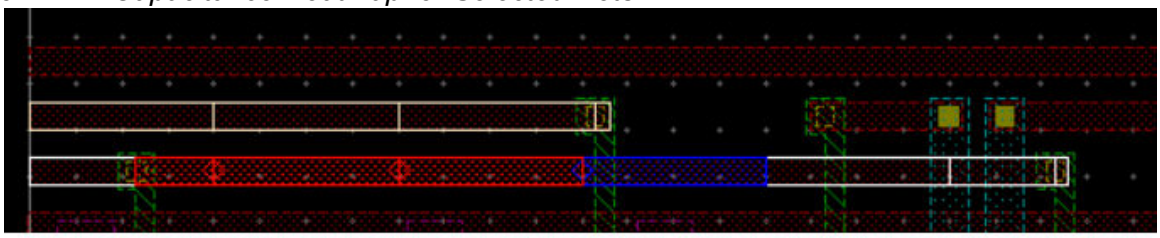
*Figure 43 Capacitance Heatmap in the Extracted View*






5. Select the required layer (for example, M1-M1). Figure 44 displays the nets on the selected layer and the diamond displays the capacitance node located. The color of the diamond indicates the coupling capacitance value on the node between net A and B.

*Figure 44 Capacitance Heatmap for Selected Nets*



The color in the extracted view of the capacitance heatmap indicates the following:

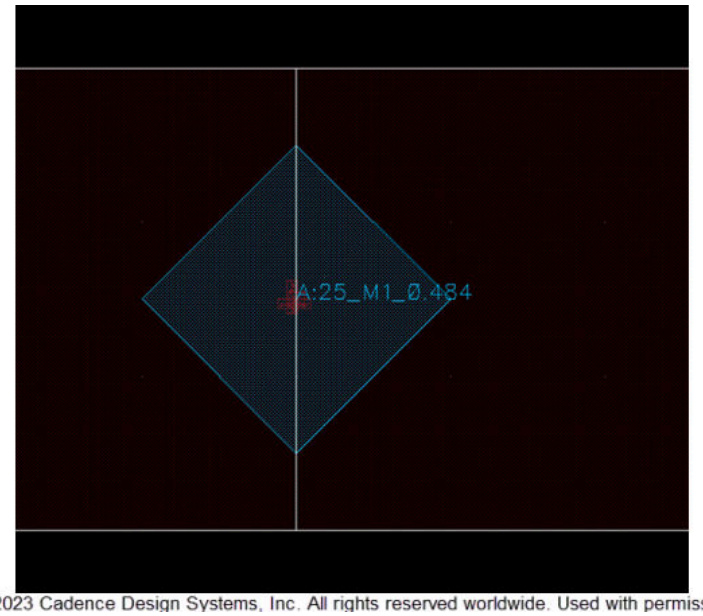
a. Red indicates highest capacitance values. If only one capacitance node is selected, then the color range is set as only red.

b. Blue indicates lowest capacitance values.

c. Green and orange indicates in-between values.

6. You can also zoom close to the diamond to view the net name, layer name, and capacitance information annotated as the **netname_layer_capvalue** as shown in Figure 45.

*Figure 45 Capacitance Heatmap Detailed View*



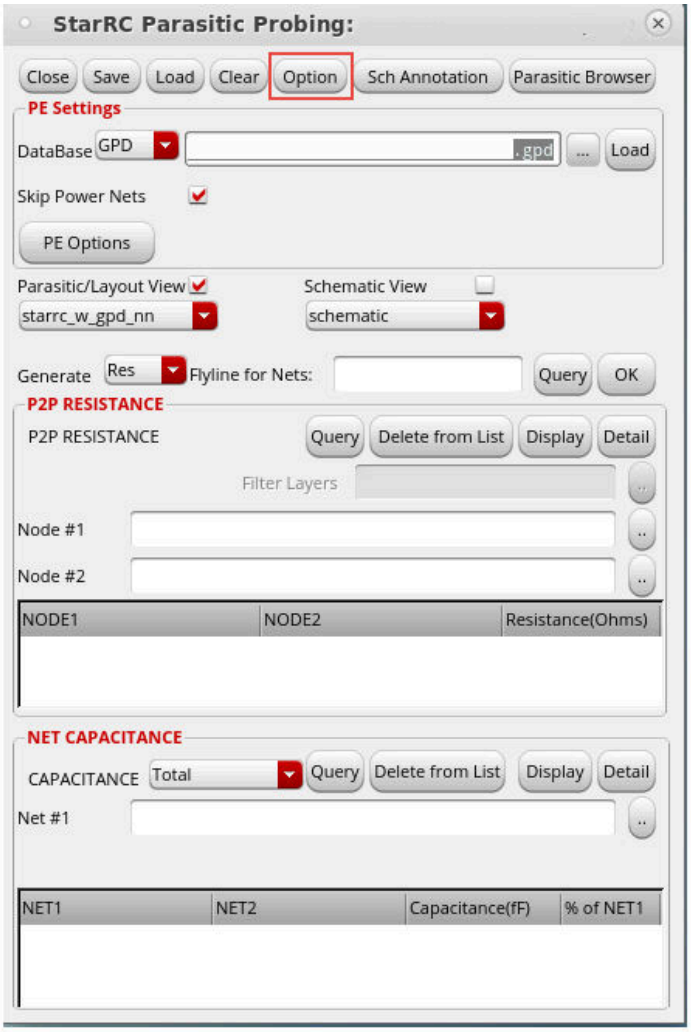
Feedback

## Changing the Annotate Font Size in Parasitic Prober

To change the annotate font size in Parasitic Prober,

1. Choose **Option** in the StarRC Parasitic Probing window (Figure 46).

*Figure 46 Changing Font Size*



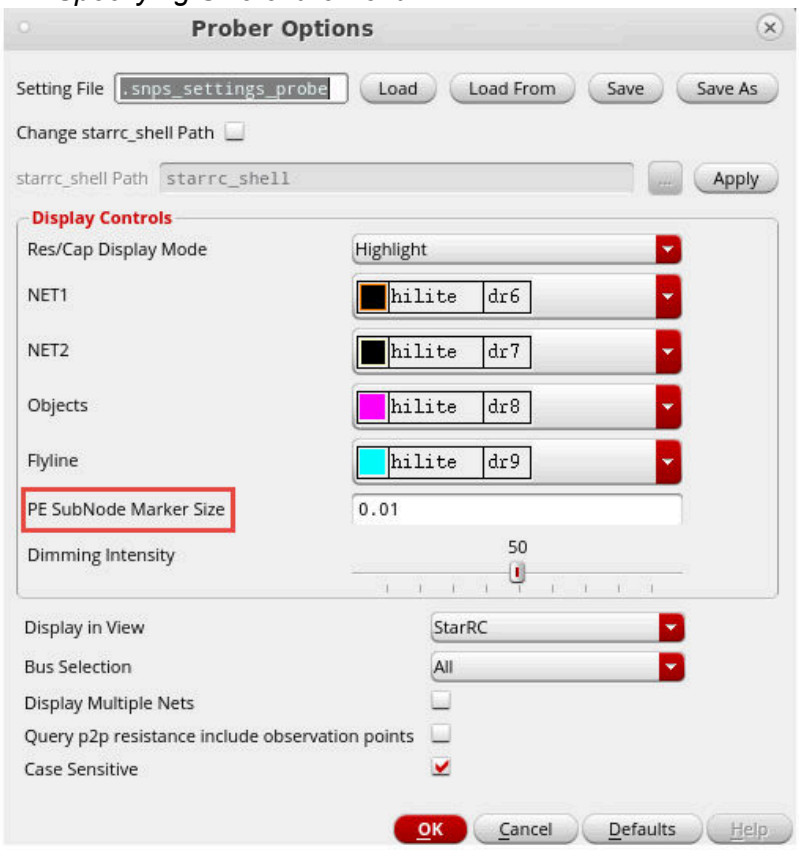



The **Probe Options** window appears (Figure 47).

2. Specify the size of the font for heatmap sub-node marker in the **PE SubNode Marker Size** field.

Feedback

*Figure 47 Specifying Size of the Font*






Note that if only one net is highlighted, the color of the net is determined by NET1. If there are two nets highlighted, the color of one net is NET1 and the color of the other net is NET2.

Feedback

# Using Tcl Commands in StarRC Shell

You can generate a report for parasitic resistors, ground capacitors, point-to-point resistance, RC contributions, and so on for specific nets using Tcl commands.

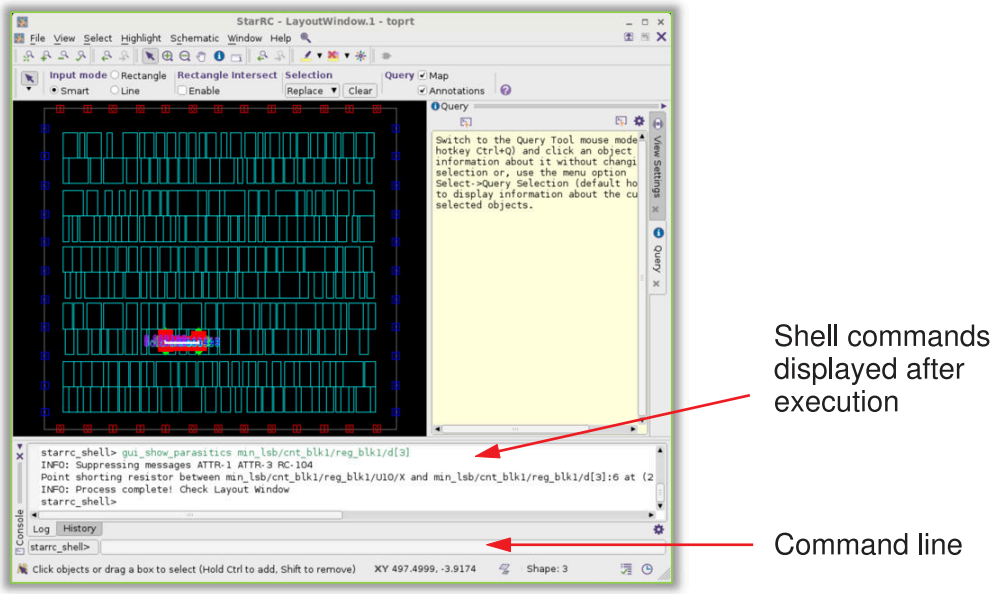
- `get*` commands such as `get_coupling_capacitors`, `get_ground_capacitors`, and `get_resistors`.
- `report*` commands such as `report_resistors`, `report_ground_capacitors`, `report_point_to_point_resistance`, and `report_rc_components`.

For more information about specific Parasitic Explorer Tcl commands, see Chapter 4, Parasitic Explorer Command Reference.

To run Tcl commands in the StarRC shell,

1. Set up the gate-level flow (see Setting Up the Gate-Level Flow and Figure 3)..

*Figure 48 GUI Console to Execute Commands*




2. Create a collection of all open nets by using the `starrc_open` net attribute:

```
starrc_shell> get_nets -filter "starrc_open==true"
{"min_lsb/cnt_blk1/n184",
"min_lsb/cnt_blk1/n191",
"min_lsb/cnt_blk1/n195"}
```

Feedback

3. Create a collection of all shorted nets by using the `starrc_short` net attribute:

```
starrc_shell> get_nets -filter "starrc_short==true"
{"sec_lsb/cnt_blk1/n157",
"sec_lsb/conv_blk1/n16"}
```

4. Report a collection object of shorts or opens. Each object is associated with an error class, which is either `open_locator` or `short`, and the object class `drc_error`.

```
starrc_shell> report_attribute -application \
           [get_drc_errors -error_data starrc_openshort.err] -nosplit

****************************
Report: Attribute
Design: toprt
Version: P-2019.03
Date: Mon Feb 11 18:23:34 2019
****************************

Design   Object    Type          Attribute       Value
-------------------------------------------------------------------------
toprt       0      string        bbox            {255.600 110.000 ...
toprt       0      collection    bounding_box    {255.600 110.000 ...
toprt       0      string        brief_info      open on net min_lsb
toprt       0      string        endpoints       {{255.600 129.200 ...
toprt       0      string        error_class     open_locator
toprt       0      collection    error_data      starrc_openshort.err
...
toprt       0      string        object_class    drc_error
...
```

5. Report a collection of error types. Each object is associated with an error class, which is either `open_locator` or `short`, and the object class `drc_error_type`.

```
starrc_shell> report_attribute -application \
           [get_drc_error_types -error_data starrc_openshort.err]

****************************
Report: Attribute
Design: toprt
Version: P-2019.03
Date: Mon Feb 11 19:03:12 2019
****************************

Design   Object          Type         Attribute      Value
-------------------------------------------------------------------------
toprt  lsb/blk1/n184   string       bbox           {255.600 110.000 ...
toprt  lsb/blk1/n184   collection   bounding_box   {255.600 110.000 ...
toprt  lsb/blk1/n184   string       brief_format   message
toprt  lsb/blk1/n184   string       brief_info     open on net min_lsb
toprt  lsb/blk1/n184   string       error_class    open_locator
toprt  lsb/blk1/n184   collection   error_data     starrc_openshort.err
```

```
...
toprt  lsb/blk1/n184   string      object_class   drc_error_type
...
```

6. Exit the StarRC shell session with the `quit` or `exit` command.

```
starrc_shell> quit
```

# 3

# Working With the Parasitic Database

The Parasitic Explorer tool provides commands to examine properties of the GPD itself.

For information about GPD commands, see the following topics:

- Querying GPD Data Stored on Disk
- Reporting GPD Properties
- Setting GPD Annotation Properties
- Getting GPD Corners and Layers
- Changing the Default Capacitance and Resistance Units

Feedback

## Querying GPD Data Stored on Disk

The Parasitic Explorer tool provides commands to examine the properties of the GPD itself. The following commands are available:

- `report_gpd_properties` – Reports the properties of the parasitic data such as completeness, the presence or absence of specific types of data, and the number of nets, cells, and ports
- `set_gpd_config` – Specifies the parasitic corners to be read and the thresholds for filtering coupling capacitors during reading
- `report_gpd_config` – Reports the option settings for reading the GPD data
- `reset_gpd_config` – Resets the settings made by the `set_gpd_config` command
- `get_gpd_corners` – Reports the parasitic corner names defined in the GPD directory
- `get_gpd_layers` – Reports the layer names defined in the GPD directory

## Reporting GPD Properties

The `report_gpd_properties` command reports general information about the data in a specified GPD directory. For example:

```
starrc_shell> report_gpd_properties -gpd MyDesignA.gpd
  ...
GPD Summary:
Properties                          Value
-------------------------------------------------------------
design_name                         MyDesignA
vendor_name                         Synopsys Inc.
program_name                        StarRC
program_version                     O-2018.06-SP4
program_timestamp                   July  1 2018 21:02:19
gpd_timestamp                       Tue Apr 10 18:26:45 2018
gpd_version                         2.6
number_of_nets                      288930
number_of_cells                     234730
...
```

The `-layers` option lists the layers present in the GPD for the specified design. For example:

```
starrc_shell> report_gpd_properties -layers -gpd MyDesignA.gpd
  ...
Layer information:
Name                Properties              Value
-----------------------------------------------------
```

```
SUBSTRATE           id                      0
SUBSTRATE           is_via                  No
poly                id                      1
poly                is_via                  No
M1                  id                      2
M1                  is_via                  No
...
```

The `-parasitic_corners` option lists the corners present in the GPD for the specified design. For example:

```
starrc_shell> report_gpd_properties -parasitic_corners -gpd MyDesignA.gpd
 ...
Corner information:
Name                Properties              Value
-----------------------------------------------------------------
CMINW125            process_name            /mydata/mypara/grd.min
CMINW125            temperature             125
CMINW125            global_temperature      25
CMINB40             process_name            /mydata/mypara/grd.min
CMINB40             temperature             -40
CMINB40             global_temperature      25
...
```

## Setting GPD Annotation Properties

The `set_gpd_config` command lets you override parameters for reading parasitic data from a GPD with the `read_parasitics -format gpd` command.

The default parameters are defined in a file called the GPD configuration file, which always exists in a GPD. You can write an ASCII version of the configuration file by using the `StarXtract -dump_gpd_config` command in the StarRC tool.

For example, the following command sets both absolute and relative thresholds for filtering coupling capacitors:

```
starrc_shell> set_gpd_config -gpd my_design1.gpd \
 -absolute_coupling_threshold 3.0e-3 \
 -relative_coupling_threshold 0.03
```

To report the GPD configuration that has been set, use the `report_gpd_config` command:

```
starrc_shell> report_gpd_config -gpd my_design.gpd
 ...

Property                          Value
-----------------------------------------------------------
absolute_coupling_threshold       0.003000
relative_coupling_threshold       0.030000
```

```
coupling_threshold_operation     and
netlist_select_nets              *
netlist_type                     {RCC *}
selected_parasitic_corners       TYP25 CWORST110 CBEST0
...
```

To include reporting of options that were set in the StarRC tool during parasitic extraction, use the `-include_starrc_options` option:

```
starrc_shell> report_gpd_config -gpd my_design.gpd
 -include_starrc_options
 ...
Property                         Value                     StarRC
-------------------------------------------------------------------------
absolute_coupling_threshold      0.003000                  N
relative_coupling_threshold      0.030000                  N
coupling_threshold_operation     and                       N
netlist_select_nets              *                         N
netlist_type                     {RCC *}                   N
selected_parasitic_corners       TYP25 CWORST110 CBEST0    N
netlist_compress                 true                      Y
dp_string                        true                      Y
netlist_connect_section          false                     Y
pin_delimiter                    /                         Y
netlist_name_map                 true                      Y
netlist_incremental              false                     Y
```

To reset options previously set by the `set_gpd_config` command, use the `reset_gpd_config` command:

```
starrc_shell> reset_gpd_config -gpd my_design.gpd
```

## Getting GPD Corners and Layers

To report the parasitic corners or layers that are present in a GPD directory, use the `get_gpd_corners` or `get_gpd_layers` command:

```
starrc_shell> get_gpd_corners -gpd my_design1.gpd
CWORST110 TYP25 CBEST0
starrc_shell> get_gpd_layers -gpd my_design1.gpd
M1 M2 M3 M4 VIA1 VIA2 VIA3
```

Feedback

# Changing the Default Capacitance and Resistance Units

The default units of capacitance and resistance are pF (farad) and kOhm respectively. You can change the units using the following variables:

- `parasitics_explorer_capacitance_unit`: Allows you to change the default capacitance unit. For example, the following command sets the capacitance unit to fF (femtofarad):

  ```
  starrc_shell> set_app_var parasitics_explorer_capacitance_unit 1e-15
  ```

- `parasitics_explorer_resistance_unit`: Allows you to change the default resistance unit. For example, the following command sets the resistance unit to ohm:

  ```
  starrc_shell> set_app_var parasitics_explorer_resistance_unit 1
  ```

To display the units used by the current design, use the `report_unit` command:

```
starrc_shell> report_unit

****************************************
Report : units
Design : simple
Version: V-2023.12
Date   : Fri Nov 17 11:55:27 2023
****************************************

Units
------------------------------------------------------------
Capacitive_load_unit        : 1e-12 Farad
Resistance_unit             : 1000 Ohm
Time_unit                   : 1e-12 Second
```

# 4

# Parasitic Explorer Command Reference

*This section provides reference information for Parasitic Explorer commands and variables.*

For more information, see the following topics:

- report_coupling_capacitors_between_nets
- report_dominant_layer_in_path
- report_ground_capacitors
- report_hierarchy
- report_instance_coordinate
- report_instances
- report_length_layerwise
- report_net_connectivity
- report_net_name
- report_nonphysical_resistors
- report_P2P_ElmoreDelay
- report_p2p_per_layer
- report_p2p_rmap
- report_parasitics_profile
- report_point_to_point_resistance
- report_ratio_aggressor_signal_coupling_to_ground_coupling
- report_ratio_coupling_from_block_to_top
- report_rcg
- report_resistors
- report_total_net_capacitance
- report_rc_components
- report_rc_corner_ratios
- report_routed_nets
- report_width_layerwise
- scale_parasitics
- set_layout_database_options
- set_power_ground_nets

- starrc_gpd_read_opens_shorts
- start_gui
- write_parasitics
- Other Supported Commands

Feedback

# check_layout_database

Reads the physical library and design files and checks the data for correctness and consistency.

### Syntax

```
check_layout_database
```

Feedback

# check_parasitics_consistency

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.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

Feedback

# current_design

Specifies the current design, which is the name used in the `BLOCK` command in the StarRC command file that is used for extraction.

### Syntax

```
current_design
  [design_name]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `design_name` | String | Specifies the design name |

### Description

Sets the current design.

You must run the `current_design` command after the `read_parasitics` command to load the GPD or SPF file.

### Examples

The following command specifies the current design name used in the `BLOCK` command:

```
starrc_shell> current_design block_name
```

### See Also

- read_parasitics

Feedback

# get_coupling_capacitors

Creates a collection of the coupling capacitors associated with one or more nets.

### Syntax

```
get_coupling_capacitors
     [-filter expression]
     [-quiet]
     [-parasitic_corners corner_names]
     [-all_parasitic_corners]
     -of_objects nets | -from node1 -to node2
     [-unit]
     [-info]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-filter_expression` | none | Refines the list of coupling capacitors by using arithmetic or relational operators with the attributes of the coupling capacitor objects. |
| `-quiet` | none | Suppresses warning and error messages if the command does not retrieve any objects. |
| `-parasitic_corners` *corner_names* | list | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-all_parasitic_corners` | none | Queries all corners in the GPD |
| `-of_objects` *nets* | list | Specifies the nets for which to return the coupling capacitors. |
| `-from` *node1* | string | Specifies a pin, port, or net internal node. The tool returns the coupling capacitors between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to` *node2* | string | Specifies a pin, port, or net internal node. The tool returns the coupling capacitors between this node and the node specified in the `-from` option. |
| `-unit` | n/a | Reports capacitance unit. |
| `-info` | n/a | Reports header information including the capacitance unit. The option is valid with the, `-of_objects` or `-from` options. |

Feedback

### Description

You can specify a node by a pin name, a port name, or a node index. To specify a node index, use the format `net_name:node_ID`. For a given net, valid node IDs are from 1 to N inclusive, where N is the number of net nodes. Note that in the GPD and in SPEF files generated from a GPD, node numbering begins at 0 and ends at N-1.

You must use either the `-of_objects` option or both the `-from` and `-to` options.

The default capacitance unit is pF.

### Examples

The following command finds the coupling capacitors attached to net abc:

```
starrc_shell> get_coupling_capacitors -of_objects abc
_sel15
```

The following command finds the coupling capacitors attached to all nets whose names begin with ABC and returns only those capacitors whose aggressor node name is XYZ:1.

```
starrc_shell> get_coupling_capacitors -of_objects ABC* \
              -filter "aggressor_node_name == XYZ:1"
_sel16
```

After the command executes, the collection handle is displayed. In the examples, `_sell5` and `_sell 6` are collection handles. The collection handle is an automatically-generated name for the collection of objects created by the command. If you want to use the objects in additional operations, you must set the collection to a variable or nest it within another command.

The following command saves the coupling capacitors of net abc into a variable named abc_cc:

```
starrc_shell> set abc_cc get_coupling_capacitors -of_objects abc
```

Use commands such as the `foreach_in_collection` command to loop through the objects in a collection. For more information about working with collections, see *Using Tcl With Synopsys Tools*.

### Attributes of Coupling Capacitors

Object properties are stored in attributes. Table 3 lists the attributes available for coupling capacitors, which have the object class `coupling_capacitor`. For coupling capacitors, the victim net is the net specified in the `get_coupling_capacitors` command. The aggressor net is the net to which the victim net is coupled by the returned parasitic capacitor.

Feedback

*Table 3 Coupling Capacitor Attributes*

| Name | Format | Definition |
|---|---|---|
| `aggressor_layer_id` | integer | The layer ID of the ITF file (nxtgrd file) for the aggressor net |
| `aggressor_layer_name` | string | The layer name of the ITF file (nxtgrd file) for the aggressor net |
| `aggressor_net` | collection | The aggressor net associated with the coupling capacitor |
| `aggressor_net_name` | string | The aggressor net name, in SPEF file format |
| `aggressor_node_ ground_ capacitor` | collection | The ground capacitor associated with the aggressor node of the coupling capacitor |
| `aggressor node_index` | integer | The index value of the node where the coupling capacitor connects to the aggressor net. Every node on a net has a unique index from 1 to N, where N is the total number of nodes on that net. |
| `aggressor_node_name` | string | The aggressor node name, in SPEF file format |
| `capacitance` | float | The single-corner capacitance value in the format used in a SPEF output file. The capacitance units are pF (different from capacitances reported in a SPEF netlist, which have units of fF). |
| `capacitance_max` | float | The maximum value of the list in the `capacitance_multicorner` attribute |
| `capacitance_min` | float | The minimum value of the list in the `capacitance_multicorner` attribute |
| `capacitance_ multicorner` | string | A list of the capacitances of all corners specified by the `-parasitic_corners` option, in the same order.<br>If the `-all_parasitic_corners` option is used, the order of the corners is the same as the order in the GPD, which is controlled by the `SELECTED_CORNERS` command in the StarRC command file used for extraction. |
| `dblayer_name` | string | The database layer name for the victim net. |
| `layer_id` | integer | The layer ID in the nxtgrd file for the victim net |
| `layer_name` | string | The ITF layer name in the nxtgrd file for the victim net. |
| `net` | collection | The victim net associated with the coupling capacitor |

Feedback

*Table 3 Coupling Capacitor Attributes (Continued)*

| Name | Format | Definition |
|---|---|---|
| `node_ground_capacitor` | collection | The ground capacitor associated with the victim node of the coupling capacitor |
| `node_index` | integer | The index value of the node where the coupling capacitor connects to the victim net |
| `node_name` | string | The victim node name, in SPEF file format |
| `object_class` | string | The value is `coupling_capacitor` |

Feedback

# get_elmore_delay

Calculates the effective Elmore delay between two nodes.

### Syntax

```
get_elmore_delay
     [-quiet corner_names]
     [-parasitic_corners corner_names]
     [-all_parasitic_corners]
     [-from node1]
     [-to node2]
     [-unit]
     [-info]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-quiet` | none | Suppresses warning and error messages if the command does not retrieve any objects |
| `-parasitic_corners` *corner_names* | list | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-all_parasitic_corners` | none | Queries all corners in the GPD |
| `-from` *node1* | string | Specifies a pin, port, or net internal node. The tool returns the coupling capacitors between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to` *node2* | string | Specifies a pin, port, or net internal node. The tool returns the coupling capacitors between this node and the node specified in the `-from` option. |
| `-unit` | n/a | Reports capacitance unit. |
| `-info` | n/a | Reports header information including the capacitance unit. The option is valid with the `-of_objects` or `-from` options. |

### Description

You can specify a node by a pin name, a port name, or a node index. To specify a node index, use the `net_name:node_ID` format. For a given net, valid node IDs are from 1 to N inclusive, where N is the number of net nodes. Note that in the GPD and in SPEF files generated from a GPD, node numbering begins at 0 and ends at N-1.

Feedback

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.

For more information, see the *Comparing the Elmore Delay* section in the *StarRC User Guide and Command Reference*.

The default time unit is picoseconds.

**Examples**

The following example calculates the Elmore delay from the my_port port to the sec/blk1/U41/my_pin pin of the my_port net for all parasitic corners:

```
starrc_shell> get_elmore_delay -from my_port -to sec/blk1/U41/my_pin
 -all_parasitic_corners
[46.2063,47.808,44.0004]
```

The following example calculates the Elmore delay from the my_port port to the sec/blk1/U41/my_pin pin of the my_port net for the typ parasitic corner:

```
starrc_shell> get_elmore_delay -from my_port -to sec/blk1/U41/my_pin
 -parasitic_corners typ
46.2063
```

Feedback

# get_ground_capacitors

Creates a collection of the ground capacitors for one or more nets.

### Syntax

```
get_ground_capacitors
     [-filter expression]
     [-quiet]
     [-parasitic_corners corner_names]
     [-all_parasitic_corners]
     -of_objects nets | -from node1 -to node2
     [-unit]
     [-info]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-filter_expression` | none | Refines the list of ground capacitors by using arithmetic or relational operators with the attributes of the ground capacitor objects. |
| `-quiet` | none | Suppresses warning and error messages if the command does not retrieve any objects. |
| `-parasitic_corners` *corner_names* | list | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-all_parasitic_corners` | none | Queries all corners that are present in the GPD. |
| `-of_objects` *nets* | list | Specifies the nets for which to retrieve the ground capacitors. |
| `-from` *node1* | string | Specifies a pin, port, or net internal node. The tool returns the ground capacitors between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to` *node2* | string | Specifies a pin, port, or net internal node. The tool returns the ground capacitors between this node and the node specified in the `-from` option. |
| `-unit` | n/a | Reports capacitance unit. |
| `-info` | n/a | Reports header information including the capacitance unit. The option is valid with the `-of_objects` or `-from` options. |

Feedback

### Description

You can specify a node by a pin name, a port name, or a node index. To specify a node index, use the format `net_name:node_ID`. For a given net, valid node IDs are from 1 to N inclusive, where N is the number of net nodes. Note that in the GPD and in SPEF files generated from a GPD, node numbering begins at 0 and ends at N-1.

You must use either the `-of_objects` option or both the `-from` and `-to` options.

The default capacitance unit is pF.

### Attributes of Ground Capacitors

Object properties are stored in attributes. Table 4 lists the attributes available for ground capacitors, which have the object class `ground_capacitor`.

*Table 4 Ground Capacitor Attributes*

| Name | Format | Definition |
|---|---|---|
| `dblayer_name` | string | The database layer name. |
| `capacitance` | float | The single-corner capacitance value, in SPEF file format. The capacitance units are pF (different from capacitances reported in a SPEF netlist, which have units of fF). |
| `capacitance_max` | float | The maximum value of the list in the `capacitance_multicorner` attribute. |
| `capacitance_min` | float | The minimum value of the list in the `capacitance_multicorner` attribute. |
| `capacitance_multicorner` | string | A list of the capacitances of all corners specified by the `-parasitic_corners` option, in the same order. If the `-all_parasitic_corners` option is used, the order of the corners is the same as the order in the GPD, which is controlled by the `SELECTED_CORNERS` command in the StarRC command file used for extraction. |
| `layer_id` | integer | The layer ID in the nxtgrd file. |
| `layer_name` | string | The ITF layer name in the nxtgrd file. |
| `net` | collection | The net that contains the ground capacitor. |
| `node_index` | integer | The index value of the node at which the ground capacitor connects to the net. Every node on a net has a unique index from 1 to N, where N is the total number of nodes on that net. |
| `node_name` | string | The node name, in SPEF file format. |

Feedback

*Table 4 Ground Capacitor Attributes (Continued)*

| Name | Format | Definition |
|---|---|---|
| `node_type` | string | The node type (`pin`, `port`, or `internal node`). |
| `object_class` | string | The value is `ground_capacitor`. |
| `x_coordinate_center` | float | The center x-coordinate (in microns) of the capacitor bounding box. |
| `x_coordinate_max` | float | The upper-right x-coordinate (in microns) of the capacitor bounding box. |
| `x_coordinate_min` | float | The lower-left x-coordinate (in microns) of the capacitor bounding box. |
| `y_coordinate_center` | float | The center y-coordinate (in microns) of the capacitor bounding box. |
| `y_coordinate_max` | float | The upper-right y-coordinate (in microns) of the capacitor bounding box. |
| `y_coordinate_min` | float | The lower-left y-coordinate (in microns) of the capacitor bounding box. |

### Examples

The following command finds the ground capacitors attached to net abc:

```
starrc_shell> get_ground_capacitors -of_objects abc
_sel15
```

The following command finds the ground capacitors attached to all nets whose names begin with ABC and returns only those capacitors whose layer ID is 12.

```
starrc_shell> get_ground_capacitors -of_objects ABC* \
               -filter "layer_id == 12"
_sel23
```

Feedback

# get_instances

Creates a collection of the instances (cells) associated with one or more nets. Valid only for transistor-level GPDs.

### Syntax

```
get_instances
     [-filter expression]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-filter` | none | Refines the list of cells by using arithmetic or relational operators with the attributes of the cell objects. |

### Description

The command checks instance or device information of a GPD parasitic database.

### Attributes of Instances

Object properties are stored in attributes. Table 5 lists the attributes available for instances.

The commands in the StarRC command file control whether some properties of cells are stored in the GPD during extraction. If the properties are not stored in the GPD, they are not available in subsequent Parasitic Explorer attribute queries.

*Table 5 Instance Attributes*

| Name | Format | Definition |
|---|---|---|
| `name` | string | The cell name, which can be controlled by the `INSTANCE_TYPE` command for layout or schematic cells names used for instances. |
| `model_name` | string | The model name of the device. |
| `length` | float | The length of a device, in microns. |
| `width` | float | The width of a device, in microns. |
| `nfin` | integer | The fin number of a device, in microns. |
| `coordinate_x` | float | The device-center-x-coordinate of the cell, in microns. |
| `coordinate_y` | float | The device-center-y-coordinate of the cell, in microns. |

Feedback

*Table 5 Instance Attributes (Continued)*

| Name | Format | Definition |
|---|---|---|
| orientation | degree | The orientation (vertical, horizonal, or non-manhattan) of the cell. |
| spice_card | | An instance type card, where the following intances are represented as follows:<br>• M for MOS<br>• R for resistor<br>• C for capacitor<br>• L for inductance<br>• J for JFET<br>• Q for BJT<br>• D for diode<br>• X for other devices |
| properties_string | | Other properties can be specified using the attribute. |

### Examples

The following examples show how to set expressions using the `get_instances -filter` command.

```
starrc_shell> get_instances -filter "width>0.2 || model_name=~*cap*"
starrc_shell> get_instances -filter "length<0.1 && nfin>5"
starrc_shell> get_instances -filter "properties_string=~*AREA*"
starrc_shell> get_instances -filter "orientation==90"
starrc_shell> get_instances -filter "name==M0"
starrc_shell> get_instances -filter ""name==\"XIsingle_cell<7>/XI58/XM0@7

starrc_shell> report_attribute -application _se14
****************************************
Report : Instances summary
Design : add4
Version: R-2020.09
Date   : Tue Aug 18 15:11:19 2020
****************************************
Design   Object    Type    Attribute Name      Value
-------------------------------------------------
add4     instance  float   coordinate_x        0.000000
add4     instance  float   coordinate_y        0.000000
add4     instance  float   length              1.000000
add4     instance  string  model_name          n
add4     instance  string  name                0/33/M1
add4     instance  int     nfin                0
add4     instance  int     orientation         0
add4     instance  string  properties_string   AD=39p AS=39p PD=32u
 PS=32u
```

```
add4     instance  string  spice_card          M
add4     instance  float   width               13.000000
```

The following example lists all instances from the parasitic file:

```
starrc_shell> get_instances
_se12
starrc_shell> sizeof_collection _sel12
208
```

The following example sets all instances in the parasitic file:

```
starrc_shell> set all_instances [get_instances]
Information: Defining new variable 'all_instances'. (CMD-041)
_sel13
starrc_shell> sizeof_collection _sel13
108
```

### See Also

- report_instances

Feedback

# get_eeq_port

Gets the information of electrically equivalent ports.

### Examples

The following example shows the electrically equivalent (EEQ) ports information.

```
starrc_shell> get_eeq_port
================   ========   ========   ============= =============
PORTS              NETNAME    LAYER      X_COORDINATE  Y_COORDINATE
================   ========   ========   ============= =============
portA              portA      M1_mask1   2.000000      11.000000
SNPS_EEQ_portA_1   portA      M5         1.000000      15.000000
SNPS_EEQ_portA_2   portA      M1_mask2   1.500000      19.000000
portC              portC      M1_mask1   1.000000      31.000000
SNPS_EEQ_portC_1   portC      M5         1.000000      45.000000
```

Feedback

# get_point_to_point_resistance

Returns the equivalent resistance of the parasitic resistors between two nodes of a net.

### Syntax

```
get_point_to_point_resistance
     [-quiet]
     [-parasitic_corners corner_names]
     [-all_parasitic_corners]
     -from node1 -to node2
     [-unit]
     [-info]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-quiet` | none | Suppresses warning and error messages if the command does not retrieve any objects. |
| `-parasitic_corners` *corner_names* | list | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. . |
| `-all_parasitic_corners` | none | Queries all corners that are present in the GPD. |
| `-from` *node1* | string | Specifies a pin, port, or net internal node as the path startpoint. You must use the `-from` and `-to` options together. |
| `-to` *node2* | string | Specifies a pin, port, or net internal node as the path endpoint. You must use the `-from` and `-to` options together. |
| `-unit` | na | Reports resistance unit. |
| `-info` | na | Reports header information including the resistance unit. The option is valid with the, `-of_objects` or `-from` options. |

### Description

You can specify a node by a pin name, a port name, or a node index. To specify a node index, use the format `net_name:node_ID`. For a given net, valid node IDs are from 1 to N inclusive, where N is the number of net nodes. Note that in the GPD and in SPEF files generated from a GPD, node numbering begins at 0 and ends at N-1.

The default resistance unit is kOhm.

Feedback

### Examples

The following example shows how to read the parasitics, find the equivalent resistance, and report the resistance value in kOhm:

```
starrc_shell> read_parasitics -keep_capacitive_coupling
    -format gpd cell.gpd
1
starrc_shell> current_design DESIGN
Loading parasitic explorer environment...
Linking design DESIGN...
Design 'DESIGN' was successfully linked.
Information: There are 28 leaf cells, ports, hiers and 5 nets in the
 design (LNK-047)
Information: Log for 'read_parasitics command' will be generated in
 'parasitics_command.log'. (PARA-107)
1

starrc_shell> get_point_to_point_resistance -from ABC/XY0/AB -to
 XYZ/XY/GATE
1.66562

starrc_shell> get_point_to_point_resistance -from ABC/XY1/DC -to
 XYZ/XY1/GATE
1.38459
```

Feedback

# get_resistors

Creates a collection of the parasitic resistors for one or more nets.

### Syntax

```
get_resistors
     [-filter expression]
     [-quiet]
     [-parasitic_corners corner_names]
     [-all_parasitic_corners]
     -of_objects nets | -from node1 -to node2
     [-shortest]
     [-unit]
     [-info]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-filter_expression` | none | Refines the list of resistors by using arithmetic or relational operators with the attributes of the resistor objects. |
| `-quiet` | none | Suppresses warning and error messages if the command does not retrieve any objects |
| `-parasitic_corners` *corner_names* | list | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-all_parasitic_corners` | none | Queries all corners that are present in the GPD |
| `-of_objects` *nets* | list | Specifies the nets for which to retrieve the parasitic resistors. |
| `-from` *node1* | string | Specifies a pin, port, or net internal node. The tool returns the parasitic resistors between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to` *node2* | string | Specifies a pin, port, or net internal node. The tool returns the parasitic resistors between this node and the node specified in the `-from` option. |
| -shortest | na | Displays the shortest path between `from_node` and `to_node` of the resistor. You must use the `-from` and `-to` options. |
| `-unit` | na | Reports resistance unit. |

Feedback

| Option and Argument | Data Type | Description |
|---|---|---|
| `-info` | na | Reports header information including the resistance unit. The option is valid with the `-of_objects` or `-from` options. |

### Description

You can specify a node by a pin name, a port name, or a node index. To specify a node index, use the format `net_name:node_ID`. For a given net, valid node IDs are from 1 to N inclusive, where N is the number of net nodes. Note that in the GPD and in SPEF files generated from a GPD, node numbering begins at 0 and ends at N-1.

You must use either the `-of_objects` option or both the `-from` and `-to` options. The `-from` and `-to` options are valid for nets that contain loops between the nodes.

The default resistance unit is kOhm.

### Attributes of Parasitic Resistors

Object properties are stored in attributes. Table 6 lists the attributes available for parasitic resistors, which have the object class `resistor`.

The commands in the StarRC command file control whether some properties of parasitic resistors are stored in the GPD during extraction. If the properties are not stored in the GPD, they are not available in subsequent Parasitic Explorer attribute queries. The following commands affect parasitic resistor attributes:

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

Feedback

- 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

*Table 6 Parasitic Resistor Attributes*

| Name | Format | Definition |
|---|---|---|
| `area` | float | The via area in square microns. Populated only if the `is_via` attribute is `true`; mutually exclusive with the `length` and `width` attributes. |
| `dblayer_name` | string | The layer name of the database layer. If resistor detail is not available in the GPD file, the `layer_id` and `layer_name` attributes are estimated using the associated ground capacitor layers. |
| `is_short` | Boolean | The value is `true` if the resistor is a shorting resistor. |
| `is_non_physical` | Boolean | The value is `true` if the resistor is a non physical resistor. |
| `is_via` | Boolean | The value is `true` if the resistor is a via resistor. |
| `is_via_array` | Boolean | The value is `true` if the resistor is part of a via array. |
| `is_via_ladder_em` | Boolean | The value is `true` if the resistor is associated with a via ladder in an NDM format IC Compiler II database that has the `is_electromigration` attribute. |
| `is_via_ladder_high_performance` | Boolean | The value is `true` if the resistor is associated with a via ladder in an NDM format IC Compiler II database that has the `is_high_performance` attribute. |
| `layer_id` | integer | The layer ID of the ITF (nxtgrd) layer. If resistor detail is not available in the GPD, the `layer_id` and `layer_name` attributes are estimated using the associated ground capacitor layers. |
| `layer_name` | string | The layer name of the ITF (nxtgrd) layer. If resistor detail is not available in the GPD, the `layer_id` and `layer_name` attributes are estimated using the associated ground capacitor layers. |

Feedback

*Table 6 Parasitic Resistor Attributes (Continued)*

| Name | Format | Definition |
|---|---|---|
| `length` | float | The resistor length, in microns. Populated along with the width attribute only if the `is_via` attribute is `false`; mutually exclusive with the `area` attribute. |
| `net` | collection | The net that contains the parasitic resistor |
| `node1_ground_capacitor` | collection | The ground capacitor associated with node 1 of the resistor |
| `node1_index` | integer | The index of node 1, one of two nodes at which the parasitic resistor connects to the net. Each node on a net has a unique index from 1 to N, where N is the total number of nodes on the net. |
| `node1_name` | string | The name of node 1, in SPEF file format |
| `node2_ground_capacitor` | collection | The ground capacitor associated with node 2 of the resistor |
| `node2_index` | integer | The index of node 2, one of two nodes at which the parasitic resistor connects to the net. |
| `node2_name` | string | The name of node 2, in SPEF file format |
| `resistance` | float | A single-corner resistance value, in SPEF file format. The resistance units are kOhms (different from resistances reported in a SPEF netlist, which have units of Ohms). |
| `resistance_max` | float | The maximum value of the list in the `resistance_multicorner` attribute |
| `resistance_min` | float | The minimum value of the list in the `resistance_multicorner` attribute |
| `resistance_multicorner` | string | If data from multiple corners is retrieved, the string contains a list of the resistances of all corners specified by the `-parasitic_corners` option, in that order. |
| `via_array_nx` | integer | In a via array, the number of vias in the X direction. Populated only if `is_via_array` is `true`. |
| `via_array_ny` | integer | In a via array, the number of vias in the Y direction. Populated only if `is_via_array` is `true`. |
| `via_array_perimeter` | float | In a via array, the perimeter in microns. Populated only if `is_via_array` is `true`. |

Feedback

*Table 6 Parasitic Resistor Attributes (Continued)*

| Name | Format | Definition |
|---|---|---|
| `width` | float | The resistor width, in microns. Populated along with the length attribute only if the `is_via` attribute is `false`; mutually exclusive with the `area` attribute. |
| `x_coordinate_max` | float | The upper-right x-coordinate (in microns) of the resistor bounding box |
| `x_coordinate_min` | float | The lower-left x-coordinate (in microns) of the resistor bounding box |
| `y_coordinate_max` | float | The upper-right y-coordinate (in microns) of the resistor bounding box |
| `y_coordinate_min` | float | The lower-left y-coordinate (in microns) of the resistor bounding box |

### Examples

For example, the following command finds the parasitic resistors attached to net abc:

```
starrc_shell> get_resistors -of_objects abc
{"resistor"}
```

The following command finds the parasitic resistors between nodes 10 and 20 of net abc:

```
starrc_shell> get_resistors -from_node abc:10 -to_node abc:20
{"resistor"}
```

Feedback

# gui_clear_parasitics

Clears parasitics annotated on a net.

### Syntax

```
gui_clear_parasitics
     [nets]
     [-all]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| *nets* | string | Nets for which to clear annotated parasitics. Can be a single net or a space-delimited list of nets inside double quotation marks. If not used, all nets are cleared. Wildcard * is supported. |
| -all | Boolean | Clears parasitic annotation for all nets; on by default. |

Feedback

# gui_show_parasitics

Highlights parasitics for a specified set of nets.

### Syntax

```
gui_show_parasitics
     [-parasitic_corners corner_name]
     [-all_parasitic_corners]
     [-aggressor_net agg_net]
     [-nores]
     [-nocg]
     [-nocc]
     [-novia]
     nets
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| *nets* | string | Nets for which to show parasitics. Can be a single net or a space-delimited list of nets inside double quotation marks; at least one net is required. Wildcard * is supported. |
| -parasitic_corners *corner_name* | string | The corners for which to display parasitic element values; can be a single corner name or a space-delimited list of corner names. |
| -all_parasitic_ corners | Boolean | Specifies to show values from all corners. |
| -aggressor_net *agg_net* | string | An aggressor net for which to show coupling capacitance |
| -nores | Boolean | Does not display parasitic resistors. |
| -nocg | Boolean | Does not display parasitic ground capacitors. |
| -nocc | Boolean | Does not display parasitic coupling capacitors. |
| -novia | Boolean | Does not display via parasitics. |

Feedback

# gui_show_short_regions

Displays noncritical polygons, including metal fill polygons, in the vicinity of a short identified by the StarRC tool during extraction.

### Syntax

```
gui_show_short_regions
     [-gpd gpd_dir]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-gpd gpd_dir` | string | The GPD generated from the StarRC extraction. The argument is the GPD directory. |

Feedback

# pe_load_parasitics

Wrapper to Load the parasitics.

### Syntax

```
pe_load_parasitics
    -format
    [-skip_pg_net]
    [-use_spf_unit]
    parasitics
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-format` | string | Specifies the DSPF netlist files and GPD files. |
| `-skip_pg_net` | n/a | Enables skip power and ground net flow. |
| `-use_spf_unit` | n/a | Sets capacitance unit to 1e-15F and resistance unit to 1Ohm. |
| `parasitics` | string | Specifies the parasitic file. |

### Examples

The following example shows how to load the parasitic file in DSPF or GPD file.

```
starrc_shell> pe_load_parasitics -format dspf star.spf
```

```
starrc_shell> pe_load_parasitics -format gpd test.gpd
```

Feedback

# read_parasitics

Reads the GPD or SPF file to annotate RC parasitics.

### Syntax

```
read_parasitics
     [-format file_format]
     [-keep_capacitive_coupling]
     [file_name]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-format` | string | Specifies the file format with parasitics information |
| `-keep_capacitive_coupling` | NA | Keeps the coupling capacitance |
| `file_name` | string | Specifies the parasitic file name |

### Description

The `read_parasitics` command reads the GPD or SPF file.

You must run the `current_design` command after the `read_parasitics` command to load the GPD or SPF file.

### Examples

The following command reads the parasitics from the GPD or SPF file:

```
starrc_shell> read_parasitics -keep_capacitive_coupling \
              -format gpd gpd_directory

starrc_shell> read_parasitics -keep_capacitive_coupling \
              -format dspf spf_file
```

### See Also

- scale_parasitics
- current_design

Feedback

# report_bounding_box

Reports the approximate bounding box of specified nets. Valid only for transistor-level GPDs.

### Syntax

```
report_bounding_box -of_objects nets
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report the bounding box. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |

### Examples

The following example shows a bounding box report.

```
starrc_shell> report_bounding_box -of_objects "SUM0 B0"
========        ===========    =========    ===========    =========
Net Name        llx            lly          urx            ury
========        ===========    =========    ===========    =========
SUM0            -467.000000    11.000000    -458.000000    82.000000
B0              -497.000000    2.500000     -272.000000    82.000000
```

Feedback

# report_compare_nets_rc

Compares resistance and capacitance of the specified nets.

### Syntax

```
report_compare_nets_rc
    -net1 net_name
    -net2 net_name
    [-output_file file_name]
    [-parasitic_corners]
    [-use_spf_unit]
    [-raw]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-net1` and `-net2` | string | Specifies net names to compare resistance and capacitance. |
| `-output_file` | string | Specifies a file name for the generated file. |
| `-parasitic_corners` | string | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-use_spf_unit` | n/a | Changes the unit for resistance and capacitance.<br>Resistance unit: Ohm (Ω)<br>Capacitance unit: femtofarad (fF) |
| `-raw` | n/a | Displays information without header titles. |

### Description

To use this command, you must specify the following commands:

- Set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.
- Set the `EXTRA_GEOMETRY_INFO: NODE` command to extract the values of the bounding boxes for nodes.

The default capacitance unit is pF and the resistance unit is kOhm. The reported layers are the database layers.

For detailed information to use the commands, see the *StarRC User Guide and Command Reference*.

Feedback

**Note:**

To use the command, contact Synopsys support center for the license information.

### Examples

The following example shows the resistance and capacitance comparison report for the specified nets.

```
starrc_shell> report_compare_nets_rc -net1 a_p -net2 a_i
************************************************************
Report              : Compare RC Components Between Net1: a_p & Net2: a_i
Design              : TEST
Version             : V-2023.12
Date                : Tue Nov 21 13:23:45 2023
Capacitive_load_unit: 1e-12 Farad
Resistance_unit     : 1000 Ohm
************************************************************

Layer Net1LayerLength Net2LayerLength Net1ResValue Net2ResValue
 ResPercentDifference Net1CapValue Net2CapValue CapPercentDifference
 Net1CCapValue Net2CCapValue
====  ============== ============== ============ ============ ====
M11 43.157600 43.157600 0.002610 0.002611 0.038300 0.008272 0.008115
 -1.934689 0.008272 0.008115
M12 21.469400 21.788500 0.000285 0.000290 1.724138 0.003144 0.003213
 2.147526 0.003144 0.003213
```

# report_compare_symmetric_nets_capacitance

Compares capacitance of the specified symmetric nets.

### Syntax

```
report_compare_symmetric_nets_capacitance
     -pattern_net1 pattern_for_net1
     -pattern_net2 pattern_for_net2
     [-nocase_sensitivity]
     [-output_file file_name]
     [-symmetric_threshold]
     [-use_spf_unit]
     [-raw]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| -pattern_net1-pattern_net2 | string | Specifies patterns for the specified symmetric nets. |
| -nocase_sensitivity | n/a | Specifies that the patterns are not case-sensitive. |
| -output_file | string | Specifies a file name for the generated file. |
| -symmetric_threshold | float | Specifies a threshold for symmetric nets to generate a report. |
| -use_spf_unit | n/a | Changes the unit for capacitance.<br>Capacitance unit: femtofarad (fF) |
| -raw | n/a | Displays information without header titles. |

### Description

The default capacitance unit is pF.

### Note:

To use the command, contact Synopsys support center for the license information.

### Examples

The following example shows the capacitance comparison report for the symmetric nets.

```
starrc_shell> report_compare_symmetric_nets_capacitance -pattern_net *in*
 -pattern_net *qn*

**************************************************
```

```
Report              : symmetric nets capacitance
Design              : TEST
Version             : V-2023.12
Date                : Tue Nov 21 13:23:45 2023
Capacitive_load_unit: 1e-12 Farad
************************************************

Net Tcap Gcap %Cg/Ct Ccap %Cc/Ct SNet STcap SGcap %SCg/SCt SCcap %SCc/SCt
 %(Tc-STc)/Tc PASS/FAIL
========  ========  ========  ========  ========  ========= ========
XMUX/in1 2.013077 0.001240 0.061597 2.011837 99.938403 XMUX/qn1 2.012006
 0.000145 0.007207 2.011861 99.992793 0.053202 PASS
XMUX/in3 8.036696 0.002748 0.034193 8.033948 99.965807 XMUX/qn3 8.034564
 0.001031 0.012832 8.033533 99.987168 0.026528 PASS
```

Feedback

# report_coupling_capacitors

Reports the coupling capacitors for specified nets.

### Syntax

```
report_coupling_capacitors
     -of_objects nets | -from node1 -to node2     [-verbose]
     -layer layer_name -net1 net_name -net2 net_name
     [-use_spf_unit]
     [-raw]
     [-parasitic_corners]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report the coupling capacitors. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |
| `-from node1` | string | Specifies a pin, port, or net internal node. The tool reports the coupling capacitors between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to node2` | string | Specifies a pin, port, or net internal node. The tool reports the coupling capacitors between this node and the node specified in the `-from` option. |
| `-verbose` | | Provides additional information about the coupling capacitors. |
| `-layer` | n/a | Reports coupling capacitance for each database layer. |
| `-net1` | list | Provides information about net1. |
| `-net2` | string | Provides information about net2. |
| `-use_spf_unit` | n/a | Changes the unit for capacitance.<br>Capacitance unit: femtofarad (fF) |
| `-raw` | n/a | Displays information without header titles. |
| `-parasitic_corners` | string | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |

### Description

You must use either the `-of_objects` option or both the `-from` and `-to` options.

Feedback

The default report contains a section for each victim net (the nets specified in the command arguments). The victim net heading lists the total coupling capacitance for the victim net. For each aggressor net, the report lists the total coupling capacitance between the aggressor net and the victim net and its percentage with respect to the total coupling capacitance on the victim net.

The verbose report also contains a section for each victim net. It provides detailed information about the individual coupling capacitances between nodes of the victim net and nodes of the aggressor nets.

For the layer report, set the `EXTRA_GEOMETRY_INFO: NODE` command to extract the values of the bounding boxes for nodes. For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

The default capacitance unit is pF.

**Examples**

The following example shows a default coupling capacitor report.

```
starrc_shell> report_coupling_capacitors -of_objects SUM0
================================
Net: SUM0
Total capacitance: 0.013721
Report Type: Aggressors, summary
================================
Total CCAP      %Cc/Ct         Aggressor Net
==========      ======         =============
0.000925        6.741491       B0
0.000908        6.617593       A0
0.000468        3.410830       CIN
================================
```

The following example shows a verbose coupling capacitor report. Net SUM0 has two pins, 0/33/M2/s and 0-/33/M-1/s, which can be determined with the `get_pins` command.

```
starrc_shell> get_pins -of [get_nets SUM0]
{"0/33/M2/s", "0/33/M1/s"}
starrc_shell> report_coupling_capacitors -of_objects SUM0 -verbose
================================
Net: SUM0
Total capacitance: 0.013721
Report Type: Aggressors, detailed
================================
Victim Node Victim Lyr Aggressor Node Aggressor Lyr Capacitance %Cc/Ct
=========== ========== ============== ============= =========== ========
0/33/M2/s   SUBSTRATE  A0:24          metal1        0.000299    2.179141
SUM0:5      metal2     A0:24          metal1        0.000047    0.342541
0/33/M2/s   SUBSTRATE  A0:25          metal1        0.000060    0.437296
SUM0:5      metal2     A0:25          metal1        0.000502    3.658625
0/33/M2/s   SUBSTRATE  B0:25          metal1        0.000370    2.696596
```

```
SUM0:4      metal2     B0:25          metal1        0.000001    0.007288
...
```

The following commands report the layer information of coupling capacitors.

```
starrc_shell> report_coupling_capacitors -layer -net1 A -net2 S
              report_coupling_capacitors -layer -net1 {A B} -net2 S
=========================================
Report : Coupling Capacitors layerinfo
Design : TEST
Version: V-2023.12
Date   : Tue Nov 21 13:23:45 2023
Capacitive_load_unit   : 1e-12 Farad
*************************************
Net1 : A
Net2 : S
Net1Layer  Net2Layer    Capacitance  %Ccp/Tccp
========   ============ ===========  =========
n_poly     M0_OD1       0.000846     1.468393
M0_PO_N    M1           0.000014     0.024300
n_poly     M0_OD2       0.003129     5.430972
n_poly     M1           0.003917     6.798695
n_poly     ndiff        0.047074     81.705835
M1         M1           0.002634     4.571805
```

Feedback

# report_coupling_capacitors_between_nets

Reports the aggressors of a net.

### Syntax

```
report_coupling_capacitors_between_nets
    -victim_net victim_net_name
    -aggressor_net aggressor_net_name
    [-verbose]
    [-output_file file_name]
    [-use_spf_unit]
    [-raw]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-victim_net` | list | Specifies coupling capacitances between nodes of the victim nets. |
| `-aggressor_net` | string | Specifies coupling capacitances between nodes of the aggressor nets. |
| `-verbose` | n/a | Provides additional information about the coupling capacitors between the specified nets. |
| `-output_file` | string | Specifies a file name for the generated file. |
| `-use_spf_unit` | n/a | Sets capacitance unit to 1e-15F and resistance unit to 1Ohm.<br>Capacitance unit: femtofarad (fF) |
| `-raw` | n/a | Displays information without header titles. |

### Description

To use the `-verbose` option, you must first set the `EXTRA_GEOMETRY_INFO:NODE` command to extract the values of the bounding boxes for nodes. The reported layers are the database layers.

The default capacitance unit is pF.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

Feedback

### Examples

The following example shows a default coupling capacitor report.

```
starrc_shell> report_coupling_capacitors_between_nets -victim_net I
 -aggressor_net S
=========================================
Report : Coupling Capacitors Between Nets
Design : TEST
Version: V-2023.12
Date   : Tue Nov 21 13:23:45 2023
Capacitive_load_unit   : 1e-12 Farad

=========================================
Total CCAP     %Cc/Ct        Aggressor Net
==========     =========     =============
0.057614       37.443783     S
=========================================
```

The following example shows a verbose coupling capacitor report.

```
starrc_shell> rreport_coupling_capacitors_between_nets -victim_net I
 -aggressor_net S -verbose
============================================================================
Report : Coupling Capacitors Between Nets
Design : TEST
Version: V-2023.12
Date   : Tue Nov 21 13:23:45 2023
Capacitive_load_unit   : 1e-12 Farad

============================================================================
Victim Node Victim Lyr Aggressor Node Aggressor Lyr Capacitance %Cc/Ct
=========== ========== ============== ============= =========== ========
I:4         n_poly     S:3            M0_OD1        0.000846    0.549822
I:4         n_poly     S:4            M0_OD2        0.003129    2.033561
I:4         n_poly     S:5            M1            0.001442    0.937167
I:4         n_poly     S:6            M1            0.001601    1.040502
I:3         M1         S:7            M1            0.002634    1.711857
I:4         n_poly     S:7            M1            0.000469    0.304807
I:5         n_poly     S:7            M1            0.000405    0.263213
I:10        M0_PO_N    S:7            M1            0.000014    0.009099
I:4         n_poly     S:34           tndiff        0.042408    27.561286
I:5         n_poly     S:34           tndiff        0.004666    3.032469
...
```

Feedback

# report_dominant_layer_in_path

Reports the layers with the most capacitance for specified nets.

### Syntax

```
report_dominant_layer_in_path
     -of_objects nets
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report the layer information. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |

### Description

To use this command, you must specify the following commands:

- Set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.
- Set the `EXTRA_GEOMETRY_INFO: NODE` command to extract the values of the bounding boxes for nodes.

For detailed information to use the commands, see the *StarRC User Guide and Command Reference*.

The default capacitance unit is pF and the resistance unit is kOhm. The reported layer is a database layer.

### Examples

The following example shows a dominant layer report.

```
starrc_shell> report_dominant_layer_in_path -of_objects "SUM0 B0"
====================================
Report              : Dominant Layer in Path
Design                  : TEST
Version                 : V-2023.12
Date                    : Tue Nov 21 13:23:45 2023
Capacitive_load_unit    : 1e-12 Farad
Resistance_unit         : 1000 Ohm

====================================
List of nets in specified timing path:
net1: SUM0
net2: B0
```

```
Total number of nets in the timing path: 2

R dominant layer: poly
Total R on poly: 0.568534

C dominant layer: metal1
Total C on metal1: 0.055679
```

Feedback

# report_ground_capacitors

The `report_ground_capacitors` command reports the ground capacitors for specified nets.

### Syntax

```
report_ground_capacitors
     -of_objects nets | -from node1 -to node2
     [-total]
     [-layer]
     [-use_spf_unit]
     [-raw]
     [-parasitic_corners]
     [-verbose]
```

You must use either the `-of_objects` option or both the `-from` and `-to` options.

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects` *nets* | list | Nets for which to report the ground capacitors. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |
| `-from` *node1* | string | Specifies a pin, port, or net internal node. The tool reports the ground capacitors between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to` *node2* | string | Specifies a pin, port, or net internal node. The tool reports the ground capacitors between this node and the node specified in the `-from` option. |
| `-total` | n/a | Reports total ground capacitance. |
| `-layer` | n/a | Reports ground capacitance for each database layer. |
| `-use_spf_unit` | n/a | Sets capacitance unit to 1e-15F and resistance unit to 1Ohm.<br>Capacitance unit: femtofarad (fF) |
| `-raw` | n/a | Displays information without header titles. |
| `-parasitic_corners` | string | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-verbose` | | Provides additional information for the ground capacitors. |

Feedback

### Description

For the layer report, set the `EXTRA_GEOMETRY_INFO: NODE` command to extract the values of the bounding boxes for nodes. For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

The default capacitance unit is pF.

### Examples

The following example shows a ground capacitor report.

```
starrc_shell> report_ground_capacitors -of_objects "SUM0 B0"
================================
Report : Ground Capacitors summary
Design : TEST
Version: V-2023.12
Date   : Tue Nov 21 13:23:45 2023
Capacitive_load_unit   : 1e-12 Farad
================================

Net: SUM0
Total capacitance: 0.013721
Report Type: Ground Capacitors
================================
Node            Layer       Capacitance       %Cc/Ct
====            =====       ===========       ======
0/33/M2/s       SUBSTRATE   0.000749          5.458786
0/33/M1/s       SUBSTRATE   0.000387          2.820494
SUM0:4          metal2      0.000247          1.800160
SUM0:5          metal2      0.002221          16.186867
...
================================
Report : Ground Capacitors summary
Design : TEST
Version: V-2023.12
Date   : Tue Nov 21 13:23:45 2023
Capacitive_load_unit   : 1e-12 Farad
================================


Net: B0
Total capacitance: 0.089779
Report Type: Ground Capacitors
================================
Node            Layer       Capacitance       %Cc/Ct
====            =====       ===========       ======
B0              metal2      0.000000          0.000000
0/38/M2/g       poly        0.000000          0.000000
0/38/M5/g       poly        0.000000          0.000000
0/54/M5/g       poly        0.000000          0.000000
...
```

```
B0:10           metal2      0.000082          0.091335
B0:11           metal2      0.001010          1.124985
```

Feedback

# report_hierarchy

Reports the reference hierarchy of the specified current instance or design.

### Syntax

```
report_hierarchy
     [-full]
     [-noleaf]
     [-nosplit]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-full` | string | Displays the full hierarchy.<br>By default, the tool lists only one time the components of submodules from multiple locations in the hierarchy. An ellipsis (...) indicate the contents of a previously displayed module. |
| `-noleaf` | string | Does not display the leaf library cells. |
| `-nosplit` | string | Does not split lines if a column overflows. |

### Description

If you set the current instance with the `report_hierarchy` command, the tool displays reference libraries of the specified instance from the design. Otherwise, the tool generates the report for the current design.

### Examples

The following command displays the report for the current design:

```
starrc_shell> report_hierarchy -full
**************************************
Report : hierarchy
Design : middle
**************************************
FD2       tech_lib
ND2       tech_lib
inter
low
NR4       tech_lib
low
NR4       tech_lib
```

Feedback

# report_instance_coordinate

Reports the bounding box of the specified instance.

### Syntax

```
report_instance_coordinate -of_objects
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects` | string | Specifies an instance name to report the bounding box information. |

### Examples

The following examples shows the `report_instances` command report:

```
starrc_shell> report_instance_coordinate -of_objects M0

**************************************************************************
Report : Instances summary
Design : add4
Version: U-2022.12-SP4
Date   : Tue May 11 15:11:19 2023

X_COORDINATE_MIN  Y_COORDINATE_MIN X_COORDINATE_MAX  Y_COORDINATE_MAX

================= ================ ================= ================
3.276000          6.331875         3.381000          6.358125
```

Feedback

# report_instances

Creates a collection of the instances (cells) associated with one or more nets. Valid only for transistor-level GPDs.

### Syntax

```
report_resistors
     [-filter expression]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-filter` | list | Nets for which to report the cells. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |

### Description

The command reports all attributes of the instances with the same format as the instance section of a SPF file. Reports instances with their name, pin or port names, model name, and their properties. Also, provides the location information of a device at the end of the report if the device location is available.

### Examples

The following examples shows the `report_instances` command report:

```
starrc_shell> report_instances -filter "width>0.2 || model_name=~*cap*"
starrc_shell> report_instances -filter "length<0.1 && nfin>5"
starrc_shell> report_instances -filter "properties_string=~*AREA*"
starrc_shell> report_instances -filter "orientation==90"
starrc_shell> report_instances -filter "name==M0"
starrc_shell> report_instances -filter
 "name==\"XIsingle_cell<7>/XI58/XM0@7\""

**************************************
Report : Instances summary
Design : add4
Version: U-2022.12-SP4
Date   : Tue May 11 15:11:19 2023

type name model_name length width nfin resistance capacitance pin_port
 properties_string
==== ==== ========== ====== ===== ==== ========== ===========
D    D1   nwdio      NA     NA    NA   NA         NA
 D1:ANODE,D1:CATHODE AREA=9.18982p PJ=101.458u
```

```
X    M0   nch_mac    0.100u 0.250u NA  NA         NA
 M0:DRN,M0:GATE,M0:SRC,M0:BULK AD=0.04975p AS=0.05025p
```

### See Also

- get_instances

Feedback

# report_length_layerwise

Reports the distribution of length with respect to database layers for specified nets.

### Syntax

```
report_length_layerwise -of_objects nets
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report lengths. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |

### Description

To use this command, you must set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

**Note:**

To use the command, contact Synopsys support center for the license information.

### Examples

The following example shows a net length report. The report contains a section for each specified net with a list of layers and the length of the specified net in each layer.

```
starrc_shell> report_length_layerwise -of_objects "SUM0 B0"
================================
Net: SUM0
Report: Layerwise length of net
================================
metal1 21u
metal2 55.5u
================================
Net: B0
Report: Layerwise length of net
================================
metal1 252.5u
metal2 113u
poly   189u
```

Feedback

# report_net_connectivity

Reports the ports, instances, and cells connected to the specified nets.

### Syntax

```
report_net_connectivity list or collection of nets
```

### Examples

The following example shows a detailed connectivity report for nets with escape characters.

```
starrc_shell> report_net_connectivity [get_nets -exact {net\\[0\\]}]
============================
Net: count_en
Report Type: Net Connectivity
============================
--------   ---------   ------------   ------------
*P Name    Direction   x-coordinate   y-coordinate
--------   ---------   ------------   ------------
count_en   in          492400.000000  400.000000
--------   ----------- ----   ------------  -----------
*I Name    Direction   Cell   x-coordinate  y-coordinate
--------   ---------   ----   ------------  -----------
U86/A      in          U86    342000.000000 254600.000000
U87/A      in          U87    319600.000000 254000.000000
---- ---------------- ---------------- ---------------- ----------------
Cell x-coordinate min y-coordinate min x-coordinate max  y-coordinate max
---- ---------------- ---------------- ---------------- ----------------
U86  342000.00         254600.00         345200.00         257200.00
U87  312600.00         254000.00         319600.00         258950.00
```

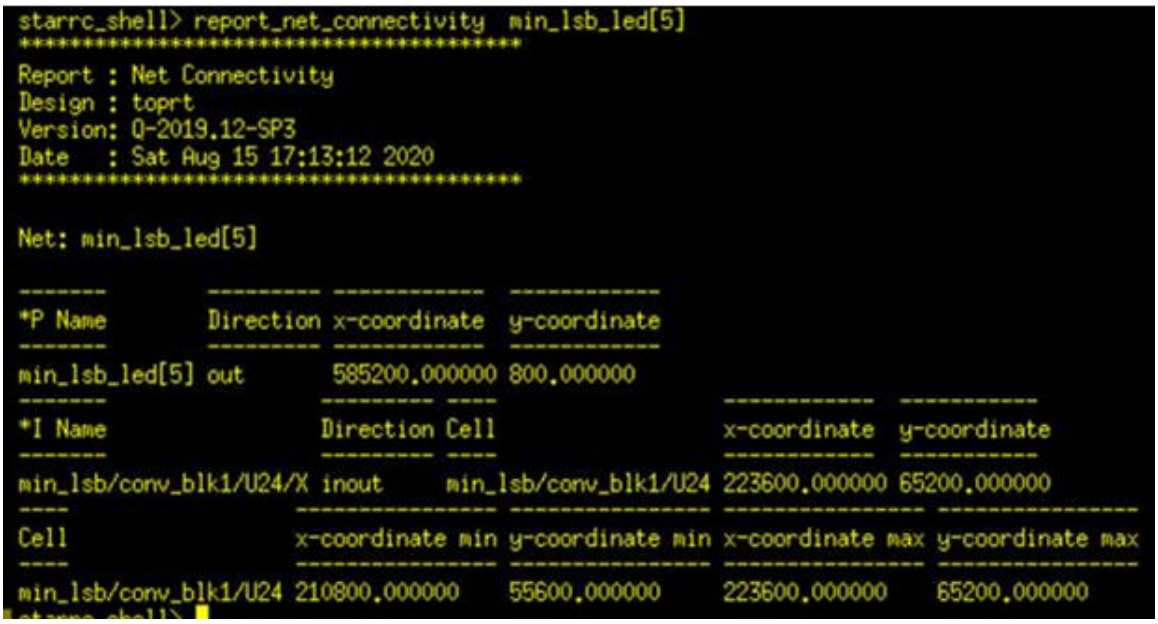
Where,

- *P report has pin names, direction, and x and y coordinates.
- *I report has port names, direction, cell name, and x and y coordinates.
- Cell report has cell names, and x and y coordinates of bounding box.

Feedback

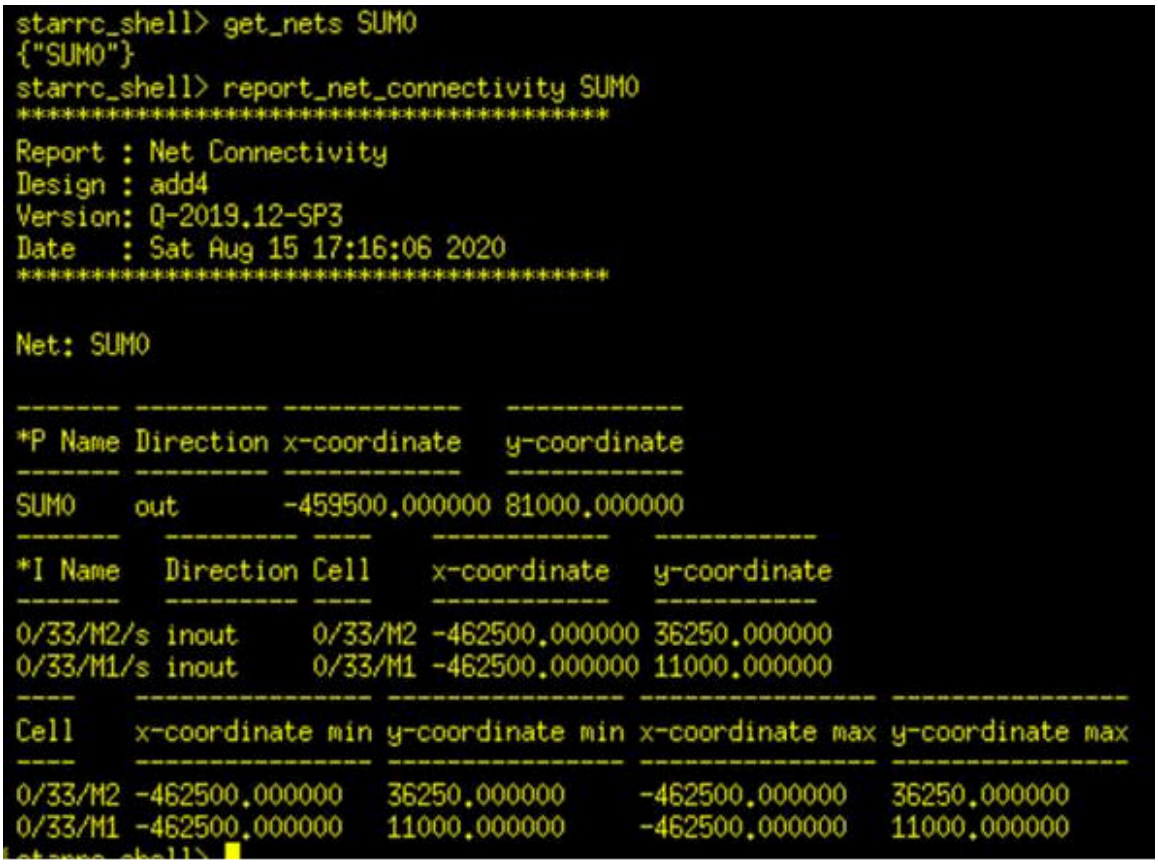
Figure 49 and Figure 50 show reports for gate-level and transistor-level GPD flows, respectively.

*Figure 49 Connectivity Report for Gate-Level Flow*



Feedback

*Figure 50 Connectivity Report for Transistor-Level Flow*



Feedback

# report_net_name

Reports net name from the parasitics based on the query requirements.

### Syntax

```
report_net_name
    [-node]node
    [-power]
    [-quiet]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-node` *node* | string | Displays the net for the specified node. |
| `-power` | n/a | Displays the power and the ground net. |
| `-quiet` | n/a | Suppresses the warning and the error messages if the command does not retrieve any objects. |

### Description

When the `-power` option is specified with the `report_net_name` command, the power nets are reported only if the skip power net flow is enabled with the `set_power_ground_nets -enable` command.

Feedback

# report_nonphysical_resistors

Reports the nonphysical resistors associated with the specified nets.

### Syntax

```
report_nonphysical_resistors
     -of_objects nets | -from node1 -to node2
     [-verbose]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report the nonphysical resistors. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |
| `-from node1` | string | Specifies a pin, port, or net internal node. The tool reports the nonphysical resistors for paths between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to node2` | string | Specifies a pin, port, or net internal node. The tool reports the nonphysical resistors for paths between this node and the node specified in the `-from` option. |
| `-verbose` | n/a | Provides additional information. |

### Description

You must use either the `-of_objects` option or both the `-from` and `-to` options.

To use this command, you must set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

The default resistance unit is kOhm.

### Examples

The following example shows a default nonphysical resistor report.

```
starrc_shell> report_nonphysical_resistors -of_objects min_lsb[5]

****************************************
Report           : Non-Physical Resistors Net Based summary
Design           : toprt
Version          : V-2023.12
```

```
Date            : Tue Nov 21 13:23:45 2023
Resistance_unit : 1000 Ohm
****************************************

Net: min_lsb[5]

Node1           Node2           Resistance
=====           =====           ==========
min_lsb/U24/X   min_lsb[5]:24   0.000001
```

The following example shows a verbose nonphysical resistor report.

```
starrc_shell> report_nonphysical_resistors -of_objects min_lsb[5] \
                                           -verbose

****************************************
Report : Non-Physical Resistors Net Based detailed
Design : toprt
Version: Q-2019.12
Date   : Mon Nov 11 17:15:12 2019
****************************************

Non-Physical Resistor Categories:
A - To connect resistively connected groups (RCGs) when physical opens
    exist in the design
B - To connect electrically equivalent nodes under specific situations
  - To short bulk nodes of MOS devices
  - Superconductive metal resistors
  - To short overlapping skip cell material
  - Very small aspect ratio resistors (l<<w)
C - Shorting resistors used on device layers in a special transistor
    level flow
D - To short pin shapes that are not explicitly connected together
E - Superconductive via resistors
F - Gate adjustment resistors (Rgdelta)
G - MOS gate delta resistors
H - To detect fuse configurations in the layout

Net: min_lsb[5]

Node1           Node2          Resistance  Layer  Length    Width      Type
=====           =====          ==========  =====  ======    =====      ====
min_lsb/U24/X   min_lsb[5]:24  0.000001     M1    0.000000 10.000000  B
```

Feedback

# report_P2P_ElmoreDelay

Reports point-to-point resistance and Elmore delay distribution information for the specified nets.

### Syntax

```
report_P2P_ElmoreDelay
    [-of_objects objects]
    [-limit]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects` | list | Reports point-to-point resistance and Elmore delay for the specified nets. |
| `-limit` | integer | Specifies the number of point-to-point pairs. |

### Description

The default resistance unit is kOhm and the default time unit is pico seconds.

### Examples

The following example calculates and reports the point-to-point resistance and Elmore delay distribution for the specified nets:

```
starrc_shell> report_P2P_ElmoreDelay -of S

************************************************
Report                          : report_P2P_ElmoreDelay
Version                         : V-2023.12
Date                            : Tue Nov 21 13:23:45 2023
Resistance_unit                 : 1000 Ohm
Time_unit                       : 1e-12 Second
************************************************

Net : S

Pin1       Pin2       P2P R          ElmoreDelay
====       ======     =========      ===========
S          M0/SRC     74.860413      0.002003
```

Feedback

# report_p2p_per_layer

Reports point-to-point resistance or Elmore delay per layer.

### Syntax

```
report_p2p_per_layer
    -from node1
    -to node2
    [-use_spf_unit]
    [-raw]
    [-parasitic_corners ]
    [-elmore_delay]
    [-skip_layers]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-from` | string | Specifies a pin, port, or net internal node as the path startpoint. |
| `-to` | string | Specifies a pin, port, or net internal node as the path endpoint. |
| `-use_spf_unit` | n/a | Changes the unit for resistance.<br>Resistance unit: Ohm (Ω) |
| `-raw` | n/a | Displays information without header titles. |
| `-parasitic_corners` | string | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-elmore_delay` | n/a | Reports Elmore delay per layer. |
| `-skip_layers` | | Specifies the layers to skip during reporting.<br>For example, `-skip_layers {M1 M2})` |

### Description

To use this command, you must:

- Set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.
- Use the `-from` and `-to` options together.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

Feedback

The default resistance unit is kOhm and the time unit is picoseconds.

**Note:**

To use the command, contact Synopsys support center for the license information.

**Examples**

The following example shows a point-to-point resistance for the specified layers.

```
starrc_shell> report_p2p_per_layer -from ABC1 -to ABC2

************************************************
Report          : point to point resistance per layer
Design          : TEST
Version         : V-2023.12
Date            : Tue Nov 21 13:23:45 2023
Resistance_unit: 1000 Ohm

************************************************

Net : I
From: I
To  : M0/GATE

Layer                    P2P_R              %P2P_R/Total
============             =========          ============
M0_PO_N                  0.000071        0.082510
M1                       0.000074        0.085049
VIA0_M0_PO_N             0.016000        18.464200
n_fptap                  0.006618        7.636765
n_poly                   0.016589        19.143900
ngate_mac                0.047303        54.587575
```

The following example shows point-to-point resistance for the specified layers with the `-elmore_delay` option.

```
starrc_shell> report_p2p_per_layer -from ABC1 -to M0/GATE -elmore_delay

************************************************
Report   : Elmore Delay per layer
Design   : TEST
Version  : U-2022.12-SP5-1-T-20231016
Date     : Wed Oct 18 11:01:06 2023
Time_unit: 1e-12 Second

************************************************

Net : I
From: I
To  : M0/GATE
```

```
Layer            ElmoreDelay     %ElmoreDelay/Total
============     =========        ============
M0_PO_N          0.000022         0.213559
M1               0.000006         0.061494
VIA0_M0_PO_N     0.001056         10.063496
n_fptap          0.000429         4.092422
n_poly           0.006302         60.082563
ngate_mac        0.002673         25.486467
```

Feedback

# report_p2p_rmap

Reports the resistive map from the given startpoint to all other points of the specified net.

### Syntax

```
report_p2p_per_layer
    -net net1
    -start_point start_point
    [-raw]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-net net_name` | string | Reports resistive map for the net. |
| `-start_point` | string | Specifies a pin, port, or net internal node as the path startpoint. |
| `-raw` | n/a | Displays information without header titles. |

### Description

The resistive map is generated as the output file. The resistive map visualization can be analyzed with the Virtuoso Integration (VI) interface. For detailed information to use the visualization, see Analyzing Parasitics Using StarRC Virtuoso Integration.

To use the `report_p2p_rmap` command, you must set the following commands:

- The `NETLIST_TAIL_COMMENTS: YES` command - Performs the original extraction and saves the information in the netlist file.
- The `EXTRA_GEOMETRY_INFO: RES NODE` - Extracts the values of the bounding boxes for nodes.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

The default resistance unit is kOhm.

**Note:**

To use the command, contact Synopsys support center for the license information.

Feedback

### Examples

The following example shows a point-to-point resistance for the resistive map.

```
starrc_shell> report_p2p_rmap -net ABC -start_point ABC

************************************************
Report : report_p2p_rmap
Design : TEST
Version: U-2022.12-SP5-VAL-20231015
Date   : Tue Oct 17 08:58:06 2023
Resistance_unit : 1000 Ohm
************************************************

Node          Resistance    X_Coordinate  Y_Coordinate  Layer
============ ==========    ============  ============  ============
A             0.000000      4.166000      56.234000     M1
D1/CATHODE    0.105578      3.670000      6.367500      nxwell
M0/DRN        0.188018      3.381000      6.345000      tndiff
A:4           0.137914      3.478000      6.227000      M0_OD1
A:5           0.123852      3.478000      6.227000      M0_OD2
```

Feedback

# report_parasitics_profile

Reports the parasitic profile, including point-to-point resistance, Elmore delay, and total capacitance of specified nets.

### Syntax

```
report_parasitics_profile
   -of_objects nets
   [-limit]
   [-tcap_thres]
   [-output_file]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects` | list | Specifies nets to include in the parasitic profile report. |
| `-limit` | integer | Specifies the number of point-to-point resistance pairs. |
| `-tcap_thres` | float | Specifies the threshold for total capacitance. |
| `-output_file` | string | Specifies a file name for the generated file. |

### Description

The default capacitance unit is pF, resistance unit is kOhm, and time unit is picoseconds.

### Examples

The following example shows the parasitics profile report.

```
starrc_shell> report_parasitics_profile -of_objects

************************************************************
Report                          : Pin1 Pin2 Netname P2P_res Elmore_delay
 Total_net_cap
Design                          : TESTVersion
Version                         : V-2023.12
Date                            : Tue Nov 21 13:23:45 2023
Capacitive_load_unit            : 1e-12 Farad
Resistance_unit                 : 1000 Ohm
Time_unit                       : 1e-12 Second
************************************************************

I       M0/GATE I       0.086656 0.009251 0.122403
```

Feedback

# report_point_to_point_resistance

Reports the point-to-point resistance for all combinations of pins and instance ports of the specified nets.

### Syntax

```
report_point_to_point_resistance
    -of_objects nets
    [-limit no_pairs]
    [-merge_parallel]
    [-parasitic_corners]
    [-raw]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report point-to-point resistance. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |
| `-limit no_pairs` | integer | Maximum number of resistances to report for each net<br>Default: 1000 |
| `-parasitic_corners` | string | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the `parasitic_corner_name` variable. |
| `-merge_parallel` | integer | Merges short pins for each net. |
| `-raw` | n/a | Displays information without header titles. |

### Description

The default resistance unit is kOhm.

### Examples

The following example shows a point-to-point resistance report.

```
starrc_shell> report_point_to_point_resistance  \
                  -of_objects min_msb/conv_blk1/n105 -limit 3

************************************************
Report          : Point to Point Resistance
Design          : toprt
Version         : V-2023.12
Date            : Tue Nov 21 13:23:45 2023
Resistance_unit: 1000 Ohm
```

```
************************************************

Net: min_msb/conv_blk1/n105

Pin1                        Pin2                        P2P R
====                        ====                        =====
min_msb/conv_blk1/U7/X      min_msb/conv_blk1/U10/A     0.025563
min_msb/conv_blk1/U7/X      min_msb/conv_blk1/U11/C     0.035199
min_msb/conv_blk1/U7/X      min_msb/conv_blk1/U14/C     0.024903

# Report without -merge_parrallel
starrc_shell> report_point_to_point_resistance  \
                  -of_objects IREF_1uA -merge_parallel

************************************************
Report : Point to Point Resistance
Version: U-2022.12-SP4
Date   : Mon May 11 12:45:00 2023
************************************************

Net: abc_lo_n

Pin1                            Pin2                            P2P R
========                        ============================    ========
IREF_1uA                        XIOSC_IREF_COMP_OPA/XM66/D       0.027471
IREF_1uA                        XIOSC_IREF_COMP_OPA/XM66@2/D     0.027471
XIOSC_IREF_COMP_OPA/XM66/D      XIOSC_IREF_COMP_OPA/XM66@2/D     0.000002

# Report with -merge_parrallel
starrc_shell> report_point_to_point_resistance  \
                  -of_objects IREF_1uA -merge_parallel

************************************************
Report : Point to Point Resistance
Version: U-2022.12-SP4
Date   : Mon May 11 12:45:00 2023
************************************************

Net: abc_lo_n
Pin1        Pin2                        P2P R
========    ========================    ========
IREF_1uA    XIOSC_IREF_COMP_OPA/XM66/D  0.027471
```

Feedback

# report_ratio_aggressor_signal_coupling_to_ground_coupling

Reports aggressors signal coupling to ground coupling ratio of a net. Valid in both gate-level and transistor-level GPD flows.

### Syntax

```
report_ratio_aggressor_signal_coupling_to_ground_coupling
    -of_objects
    [-supply_nets]
    [-ts_signal_nets]
    [-shielding_net]
    [-power_ratio]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects` | list | Specifies input for target nets. |
| `-supply_nets` | list | Specifies supply nets.<br>Default: `VDD*` |
| `-ts_signal_net` | list | Specifies a set of wire nets of thermal sensitive (TS) signal nets. |
| `-shielding_net` | list | Specifies a shielding net name.<br>Default : `VSS` |
| `-power_ratio` | float | Specifies the power net ratio.<br>Default: `0.1` |

### Description

The aggressor signal coupling to ratio of a net is calculated as follows:

$$\text{ratio} = \frac{(\text{cc\_power} \times \text{power\_ratio} + \text{cc\_signal})}{\text{cc\_shielding}}$$

where

- `CC_power`: Coupling capacitance of the target thermal sensitive net to the PG nets.
- `CC_signal`: Coupling capacitance of the target thermal sensitive net to signal nets other than the thermal sensitive nets.
- `CC_shielding`: Coupling capacitance of the target thermal sensitive net to a `VSS` net that shields the thermal sensitive nets.

Feedback

**Note:**

To use the command, contact Synopsys support center for the license information.

### Examples

The following example shows a default coupling capacitor report.

```
starrc_shell> report_ratio_aggressor_signal_coupling_to_ground_coupling /
    -of_objects B0 -ts_signal_nets A0 -shielding_net VSS -supply_nets VDD
================================
Ratio of bad cap to good cap on net : B0
0.086%

Capacitance details (unit: 1e-15 Farad):
Cg: 0.9616
CC_power: 5.6191
CC_signal: 1.5951
CC_ts_signal: 15.2488
CC_shielding: 25.0823
```

Feedback

# report_ratio_coupling_from_block_to_top

Reports coupling ratio from the block to top for nets.

### Syntax

```
report_ratio_coupling_from_block_to_top
[-of_objects nets]
[-block_name_for_port_net_inclusion]
[-replace_netname_string_of]
[-replace_netname_string_with]
[-output_coupling_report_file]
[-output_coupling_report_external_to_block_file]
[-ratio_coupling_correlation_output_report]
[-input_coupling_report_file]
[-input_file_for_victim_net_filtering]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| -of_objects | list | Specifies the input for target nets. |
| -block_name_for_port_net_inclusion | string | Specifies the block name for port net inclusion. |
| -replace_netname_string_of | string | Replaces the hierarchical net name and adds a prefix net name if available. |
| -replace_netname_string_with | string | Replaces the hierarchical net name and adds a prefix net name if available. |
| -output_coupling_report_file | string | Specifies the file name for output coupling report. |
| -output_coupling_report_external_to_block_file | string | Specifies the file name for the coupling_report_external_to_block. |
| -ratio_coupling_correlation_output_report | string | Specifies the file name for the correlation report. |
| -input_coupling_report_file | string | Specifies the file name for the input coupling report. |
| -input_file_for_victim_net_filtering | string | Specifies the file name for the victim net filtering. |

### Examples

The following example shows the report coupling ratio from block-to-top for nets.

```
starrc_shell> report_ratio_coupling_from_block_to_top -of_objects
 {XI0/*} /
```

```
   -output_coupling_report_file outfile_report
-block_name_for_port_net_inclusion XI0 /
   -replace_netname_string_of {XI0/} -replace_netname_string_with {""} /
   -input_coupling_report_file block.rpt /
   -ratio_coupling_correlation_output_report ratio.rpt /
   -output_coupling_report_external_to_block_file exteral_report
```

Feedback

# report_rcg

Reports the resistively connected group (RCG) for the specified net.

### Syntax

```
report_rcg
     -net netname
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-net` *netname* | string | Reports RCG for the net. |

### Description

To use this command, you must:

- Set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.
- Set the `EXTRA_GEOMETRY_INFO: RES NODE` command to extract the values of the bounding boxes for nodes.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

**Note:**

To use the command, contact Synopsys support center for the license information.

### Examples

The following example shows the number of resistively connected group (RCG) reports.

```
starrc_shell> report_rcg -net SUM1
================================
Report : report_rcg
Version: U-2022.12-SP5-VAL-20231006
Date   : Sun Oct  8 09:07:41 2023
================================
Info: Net: SUM1  (2 RCGs)
```

Feedback

# report_resistors

Reports the parasitic resistors for specified nets.

### Syntax

```
report_resistors
     -of_objects  | -from node1 -to node2     [-verbose]
     [-use_spf_unit]
     [-raw]
     [-shortest]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects` *nets* | list | Nets for which to report the parasitic resistors. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |
| `-from` *node1* | string | Specifies a pin, port, or net internal node. The tool reports the parasitic resistors between this node and the node specified in the `-to` option, which must both belong to the same net. |
| `-to` *node2* | string | Specifies a pin, port, or net internal node. The tool reports the parasitic resistors between this node and the node specified in the `-from` option. |
| `-verbose` | n/a | Provides additional information. |
| `-use_spf_unit` | n/a | Changes the unit for resistance.<br>Resistance unit: Ohm (Ω) |
| `-raw` | n/a | Displays information without header titles. |
| `-shortest` | na | Displays the shortest path between the `from_node` and the `to_node` of the resistor. |

### Description

The parasitic resistor report is based on either nets (using the `-of_objects` option) or on paths (using the `-from` and `-to` options).

You must use either the `-of_objects` option or both the `-from` and `-to` options.

The `-shortest` option is valid when the `-from` and `-to` options are used. Shortest path visualization can be analyzed with the Virtuoso Integration (VI).

The default resistance unit is kOhm.

Feedback

### Examples

The following example shows a path-based parasitic resistor report.

```
starrc_shell> report_resistors -from 0/33/M2/s -to SUM0:12
================================
Net: SUM0
From: 0/33/M2/s
To: SUM0:12
Report Type: Resistors, Path Based, summary
================================
Node1           Node2          Resistance
=====           =====          ==========
0/33/M1/s       SUM0:15        0.000550
SUM0:12         SUM0:14        0.000031
SUM0:13         SUM0:14        0.000237
SUM0:13         SUM0:16        0.000031
...
```

The following example shows a net-based parasitic resistor report.

```
starrc_shell> report_resistors -of_objects "SUM0 B0"
================================
Net: SUM0
Report Type: Resistors, Net Based, summary
================================
Node1           Node2          Resistance
=====           =====          ==========
SUM0            SUM0:6         0.000357
SUM0            SUM0:7         0.000031
0/33/M2/s       SUM0:11        0.000550
0/33/M1/s       SUM0:15        0.000550
SUM0:4          SUM0:5         0.000620
...
================================
Net: B0
Report Type: Resistors, Net Based, summary
================================
Node1           Node2          Resistance
=====           =====          ==========
B0:10           B0:11          0.000229
...
```

The following example shows a verbose parasitic resistor report. Values for the resistor length and width are available only if the extraction was performed with the `NETLIST_TAIL_COMMENTS` command set to `YES`. Values for the bounding box coordinates are available only if the extraction was performed with the `EXTRA_GEOMETRY_INFO` command set to `RES`.

```
starrc_shell> report_resistors -of_objects SUM0 -verbose
================================
Net: SUM0
```

Feedback

```
Report Type: Resistors, Net Based, detailed
================================
Node1       Node2    Resistance Layer     Length Width Area llx lly urx ury
=====       =====    ========== =====     ====== ===== ==== === === === ===
SUM0        SUM0:6   0.000357   metal2      NA     NA    NA   NA  NA  NA  NA
SUM0        SUM0:7   0.000031   metal2      NA     NA    NA   NA  NA  NA  NA
0/33/M2/s   SUM0:11  0.000550   SUBSTRATE   NA     NA    NA   NA  NA  NA  NA
0/33/M1/s   SUM0:15  0.000550   SUBSTRATE   NA     NA    NA   NA  NA  NA  NA
SUM0:4      SUM0:5   0.000620   metal2      NA     NA    NA   NA  NA  NA  NA
...
```

Feedback

# report_total_net_capacitance

Reports the total capacitance of the specified nets.

### Syntax

```
report_total_net_capacitance nets
     [-nets net_name]
     [-name_sort]
     [-layer layer1]
     [-skip_power_nets]
     [-use_spf_unit]
     [-raw]
     [-parasitic_corners]
     [-verbose]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| *nets* | list | Nets for which to report the total capacitance. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |
| *-name_sort* | n/a | Sorts the capacitance results based on the layer name. |
| -layer | n/a | Reports total capacitance for each database layer. |
| -skip_power_nets | n/a | Does not report power nets; wildcards are supported. |
| -use_spf_unit | n/a | Changes the unit for capacitance.<br>Capacitance unit: femtofarad (fF) |
| -raw | n/a | Displays information without header titles. |
| -parasitic_corners | string | Specifies the corners in the GPD file to query. If this option is not specified, it selects the corner specified with the parasitic_corner_name variable. |
| -verbose | | Provides additional information about the coupling capacitors. |

### Description

For the layer report, set the EXTRA_GEOMETRY_INFO: NODE command to extract the values of the bounding boxes for nodes. For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

The default capacitance unit is pF.

Feedback

### Examples

The following example shows a total net capacitance report.

```
starrc_shell> report_total_net_capacitance A -layer
************************************
Report : Total Capacitance
Design : toprt
Version: V-2023.12
Date   : Tue Nov 21 13:23:45 2023
Capacitive_load_unit : 1e-12 Farad
************************************
========    =================  =================
Layer Name  Total Capacitance  %(Cg+Cc)/Ct
========    =================  =================
M1          0.001354           46.916147
M0_OD1      0.000827           28.655579
M0_OD2      0.000657           22.765073
tndiff      0.000047           1.628552
M0_STI1     0.000001           0.034650
nxwell      0.000000           0.000000
M0_STI2     0.000000           0.000000
```

Feedback

# report_rc_components

Reports the RC contributions of the specified nets.

### Syntax

```
report_rc_components -of_objects nets
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report the RC components. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |

### Description

To use this command, you must specify the following commands:

- Set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.
- Set the `EXTRA_GEOMETRY_INFO: NODE` command to extract the values of the bounding boxes for nodes.

For detailed information to use the commands, see the *StarRC User Guide and Command Reference*.

The default capacitance unit is pF and the resistance unit is kOhm.

### Examples

The following example shows an RC component report. The report contains a section for each specified net.

```
starrc_shell> report_rc_components -of_objects min_lsb_led[5]
**********************************
Report                : RC Components
Design                : toprt
Version               : V-2023.12
Date                  : Tue Nov 21 13:23:45 2023
Capacitive_load_unit  : 1e-15 Farad
Resistance_unit       : 1000 Ohm
**********************************

Net: min_lsb_led[5]
Total Resistance : 0.172925
Total Capacitance: 0.153868
```

Feedback

```
Layer           ResValue          %ResContribution  CapValue(pF)
 %CapContribution
=====           ===============   ================  ============
 ================
M0_PO_N         0.004579          2.647969          0.002213      1.438246
M1              0.000279          0.161342          0.021919      14.245327
VIA0_M0_PO_N    0.016000          9.252566          0.000000      0.000000
n_fptap         0.006618          3.827093          0.000000      0.000000
n_poly          0.050843          29.401764         0.129736      84.316427
ngate_mac       0.094606          54.709267         0.000000      0.000000
```

Feedback

# report_rc_corner_ratios

Reports the nets with the largest ratio of parasitics between corners. Valid only for transistor-level GPDs.

### Syntax

```
report_rc_corner_ratios -parasitic_corners corner1 corner2
     [-type object]
     [-nworst n1]
     [-nbest n2]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-parasitic_corners` *corner1 corner2* | string | Two corners for which to report the nets with the largest capacitance ratio. |
| `-type` *object* | string | Specifies the type of parasitic object to compare. Valid values are `ground_capacitor`, `coupling_capacitor`, `resistor`. The default is `ground_capacitor`. |
| `-nworst` *n1* | integer | Specifies to report the largest ratios and the number of nets to report. |
| `-nbest` *n2* | integer | Specifies to report the smallest ratios and the number of nets to report. |

### Description

If neither the `-nbest` or `-nworst` options is specified, the default is `-nworst 100`.

For each net, the ratio is the capacitance (or resistance) for corner 1 divided by the capacitance (or resistance) for corner 2.

### Examples

The following example shows a coupling capacitance report for corners typ and max, listing the 100 worst ratios.

```
starrc_shell> report_rc_corner_ratios -parasitic_corners {typ max}\
                      -type coupling_capacitor
************************************
Report : RC Corner Ratios
Design : toprt
Version: Q-2019.12
Date   : Wed Nov 13 8:12:22 2019
************************************
```

```
Parasitic Corner 1: typ
Parasitic Corner 2: max
Reporting 100 nworst nets with coupling_capacitor variation between
 corners

Net                      Ratio
===                      =====
sec_msb/cnt_blk1/n181    1.054146
min_msb/cnt_blk1/n224    1.052174
min_msb_led[3]           1.051689
min_msb/conv_blk1/n122   1.050078
...
```

The following example shows a coupling capacitance report, listing only the 10 best ratios.

```
starrc_shell> report_rc_corner_ratios -parasitic_corners {typ max}\
                        -type coupling_capacitor -nbest 10
************************************
Report : RC Corner Ratios
Design : toprt
Version: Q-2019.12
Date   : Wed Nov 13 9:18:02 2019
************************************

Parasitic Corner 1: typ
Parasitic Corner 2: max
Reporting 10 nbest nets with coupling_capacitor variation between corners

Net                      Ratio
===                      =====
min_msb/conv_blk1/n107   0.903774
min_msb/conv_blk1/n125   0.904806
min_lsb_led[0]           0.905907
min_lsb/conv_blk1/n89    0.906986
sec_msb/conv_blk1/n54    0.907177
min_lsb/cnt_blk1/n196    0.908030
min_lsb/conv_blk1/n97    0.908989
min_lsb_lef[4]           0.909031
min_msb/n128             0.909055
sec_msb/conv_blk1/n50    0.909223
-------------------------------------------------
```

Feedback

# report_routed_nets

Reports the nets routed on specified database layers and the total capacitance values of those nets.

### Syntax

```
report_routed_nets -layer layers
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-layer layers` | list | Layers for which to report the routed nets. Can be a single layer or a space-delimited list of layers inside double quotation marks. Wildcard * is supported. |

### Description

To use this command, you must specify the following commands:

- Set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the required information in the netlist file.
- Set the `EXTRA_GEOMETRY_INFO: NODE` command to extract the values of the bounding boxes for nodes.

For detailed information to use the commands, see the *StarRC User Guide and Command Reference*.

### Examples

The following example shows a routed net report. The report contains a section for each specified layer with a list of nets, the total capacitance for each net, and the percentage contribution of the layer to the total capacitance.

```
starrc_shell> report_routed_nets "metal2 metal3"
==========================================
Total number of nets routed on metal2: 16
==========================================
Net Name    Total Capacitance    metal2 Capacitance    %metal2/Ct
========    =================    ==================    ==========
A0          0.092668             0.012534              13.525705
A1          0.092722             0.012547              13.531848
A2          0.092721             0.012547              13.531994
A3          0.092718             0.012544              13.529196
B0          0.089779             0.012808              14.266142
...
```

Feedback

# report_width_layerwise

Reports the distribution of all the widths with respect to the layers of the specified nets. Also, reports the corresponding length by adding the resistor lengths for a particular width on a layer.

### Syntax

```
report_width_layerwise -of_objects nets
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-of_objects nets` | list | Nets for which to report lengths. The option can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |

### Description

You must set the `NETLIST_TAIL_COMMENTS: YES` command to perform the original extraction and save the information in the netlist file.

For detailed information to use the command, see the *StarRC User Guide and Command Reference*.

**Note:**

To use the command, contact Synopsys support center for the license information.

### Examples

The following example shows a net width report. The report contains a section for each specified net with a list of layers and the width of the specified net in each layer.

```
starrc_shell> report_width_layerwise -of_objects "net83"
================================
Report : Layerwise width of net
Design : TEST
Version  : V-2023.12
Date     : Mon Oct 30 10:03:26 2023
================================
Layer              Width              Length per Width
============       =========          ============
M1                 3u                 297.5u
M1                 5u                 60u
M2                 4u                 150u
fpoly              2u                 29u
fpoly              4u                 8u
```

```
ngate                2u                   6u
nsd                  1.5u                 46u
nsd                  3u                   17u
nsd                  4.25u                12u
pgate                2u                   12u
psd                  1.5u                 58u
psd                  3u                   51u
psd                  4.25u                24u
```

# scale_parasitics

Uses the Parasitic Explorer tool commands to scale parasitics.

### Syntax

```
scale_parasitics
    [-config file_name]
    [-wildcard YES | NO]
    [-case_sensitive YES | NO]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-config file_name` | string | The file includes the syntax of options and arguments to specify the scaling factors and required information for the scaling resistance and capacitance. |
| `-wildcard` | string | Use wildcards in net names.<br>Default: `YES` |
| `-case_sensitive` | string | Specifies if the net names are case-sensitive during selection.<br>Default: `YES` |

The following options and arguments should be included in the configuration file specified with the `-config` option:

| | | |
|---|---|---|
| `-net_list net_name` | string | Lists the names of nets.<br>If you use the `-from` and `-to` options to specify a pin, port, or net, the tool ignores the nets specified with the `-net_list` option. |
| `-res_factor resistance_factor` | float | Specifies the scaling factor for resistance. If the scale factor is not specified, the tool sets the scale factor to 1. |
| `-cc_factor coupling_cap_factor` | float | Specifies the scaling factor for coupling capacitance. If the scale factor is not specified, the tool sets the scale factor to 1. |
| `-gc_factor ground_cap_factor` | float | Specifies the scaling factor for ground capacitance. If the scale factor is not specified, the tool sets the scale factor to 1. |
| `-from node1` | string | Specifies the startpoint, which is a pin, port, or net internal node. The tool returns the coupling capacitors between this node and the node specified with the `-to` option; both nodes should belong to the same net. |

Feedback

| Option and Argument | Data Type | Description |
|---|---|---|
| `-to` *node2* | string | Specifies the endpoint, which is a pin, port, or net internal node. The tool returns the coupling capacitors between this node and the node specified with the `-from` option; both nodes should belong to the same net. |
| `-layer` *layer_name* | string | Specifies the database layer name for mapping design layers.<br>If you use the `-from` and `-to` options to specify a pin, port, or net, the tool ignores the layers specified with the `-net_list` option. |
| `corner_start` *corner_name* | string | Specifies a corner to scale resistance and capacitance for the specified corner. |
| `corner_start *` | string | Specifies a corner as a startpoint to scale resistance and capacitance for all corners. |
| `-device` *instance_name* | n/a | Specifies an instance name to scale device properties. |
| `-device_parallel` *num* | integer | Removes the number of multifinger devices specified. Supports an SPF file output netlist. |

### Description

The `scale_parasitics` command allows you to specify a configuration file to scale the scaling factors for resistance and capacitance, based on nets, point-to-point resistance, capacitance, and layers. The scaled nets, corners, instances, and warning messages are saved in the scale_parasitics.log file.

Note that temperature sensitivity analysis, layer mapping, and tail comments are not supported with RC scale factors during simultaneous multicorner (SMC) extraction.

The following example shows the syntax within the configuration file:

```
-net_list net_name -res_factor res_factor -cc_factor cc_factor \
-gc_factor gc_factor -target_width res_width
-net_list net name -res_factor res_factor2 -cc_factor cc_factor2 \
-gc_factor gc_factor2 -target_width res_width
-net_list net name -from pin1/port1/node1 -to pin2/port2/node2 \
-res_factor res_factor -cc_factor cc_factor -gc_factor gc_factor

-layer layer_name -res_factor res_factor -cc_factor cc_factor \
-gc_factor gc_factor -target_width res_width
-net_list net_name -layer layer_name -res_factor res_factor \
-cc_factor cc_factor -gc_factor gc_factor -target_width res_width

corner_start corner_name
```

Feedback

```
  -net_list net_name -res_factor res_factor -cc_factor cc_factor \
  -gc_factor gc_factor -target_width res_width
  -net_list net name -from pin1/port1/node1 -to pin2/port2/node2 \
  -res_factor res_factor
  -cc_factor cc_factor -gc_factor gc_factor
  -layer layer_name -res_factor res_factor -cc_factor cc_factor \
  -gc_factor gc_factor -target_width res_width
corner_end

corner_start *
  -net_list net_name -res_factor res_factor -cc_factor cc_factor \
  -gc_factor gc_factor -target_width res_width
  -net_list net name -from pin1/port1/node1 -to pin2/port2/node2
 -res_factor res_factor \
  -cc_factor cc_factor -gc_factor gc_factor
  -layer layer_name -res_factor res_factor -cc_factor cc_factor \
  -gc_factor gc_factor -target_width res_width
corner_end
-device instance_name -w new_w_data -nf new_nf_data -m new_m_data \
   -model new_model_name -l new_l_data -cellw new_cellw_data \
   -device_parallel num
```

Consider the following rules when specifying the scaling factors:

- When you specify multiple scaling settings for resistances and capacitances, the tool performs scaling only one time on the specified nets and avoids scaling subsequently. The tool issues a warning message to indicate that the setting of resistances and capacitances are ignored.
- When you set the scale factor for coupling capacitance on a selected net, the coupling capacitance of aggressor nets are also scaled.
- When you set the scale factor for point-to-point resistor, the tool applies the scale factor on all resistors specified with the `get_resistors -from node1 -to node2` command.
- When you set the scale factor for point-to-point capacitor, the tool applies the scale factor on the capacitance with one node in the resistors specified with the `get_resistors -from node1 -to node2` command.
- When you set the precedence of modifiers used with `res_factor`, the scaling depends on the following `res_factor` sequence:
  - If `-target_width` is defined before `-layer` in a different configuration line, the `-target_width` is scaled first, and the other unscaled resistors corresponding to `-layer` are then scaled.

    For example,

    ```
    -net_list WL[3] -target_width 0.0140 -res_factor 10
    -net_list WL[3] -layer M1 -res_factor 5
    ```

- If `-target_width` and `-layer` are in the same configuration line, the shapes that satisfy `-target_width` and `-layer` at the same time are scaled.

  For example,

  ```
  -net_list WL[3] -target_width 0.0140 -layer M1 -res_factor 10
  ```

- If the resistors of a net are scaled by the `-layer` or the `-target_width` options first, and then scaled for the complete net in the configuration file, then the condition is not regarded as a conflict. As the `-layer` or the `-target_width` options scales partial resistors of the net, the configuration of scaling the complete net scales other unscaling resistors.

  For example,

  ```
  -net_list WL[3] -target_width 0.0140 -res_factor 10
  -net_list WL[3] -layer M1 -res_factor 5
  -net_list WL[3] -res_factor 20
  ```

- The following situation is considered as a conflict. In this case no `res_factor` conflicts are reported when the first one is used, and the `res_factor` of the scaled net is reported in the scale_parasitics.log file.

  For example,

  ```
  -net_list WL[3] -res_factor 10
  -net_list WL[3] -res_factor 20
  ```

**Note:**

To use the command, contact Synopsys support center for the license information.

### Examples

The following example shows how to use the `scale_parasitics` and `write_parasitics` commands:

```
# Use the Parasitic Explorer commands to specify scale factors for the
 parasitics.
starrc_shell> scale_parasitics -config

# Use the Parasitic Explorer commands to write out the GPD file after RC
 scaling.
starrc_shell> write_parasitics -pe -format gpd test.gpd
```

### See Also

- write_parasitics
- Setting Up the Gate-Level Flow

Feedback

# set_layout_database_options

Specifies options for loading the layout of the design to be analyzed.

### Syntax

```
set_layout_database_options
     [-physical_tech_lib_path file_name_list]
     [-physical_lib_path file_name_list]
     [-physical_design_path file_name_list]
     [-physical_icc2_lib lib_dir_path]
     [-physical_icc2_blocks block_name_list]
     [-physical_enable_clock_data]
     [-physical_enable_all_vias]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-physical_tech_lib_path` *file_name_list* | list | For LEF/DEF designs, a list of LEF technology files. The technology LEF files are used to understand the physical constraints, including layer definitions, via definitions, via rules, and overlapped layer constructs. If multiple LEF files are in use, specify the technology LEF files before the cell LEF files. |
| `-physical_lib_path` *file_name_list* | list | For LEF/DEF designs, a list of LEF library files. The physical library data is used to understand physical constraints such as the shapes of pins, cells, and blocks. |
| `-physical_design_path` *file_name_list* | list | For LEF/DEF designs, a list of DEF physical data files. The data is used to understand the layout details such as available free sites and utilization density. |
| `-physical_icc2_lib` *lib_dir_path* | string | For NDM format IC Compiler II designs, the reference library directory path. All technology and cell LEF information is obtained from reference libraries in this directory.<br>You can specify additional LEF files with the `-physical_lib_path` option.<br>Only one reference library directory can be specified with this option. To use more than one reference library directory, use multiple instances of the `set_layout_database_options` command. |
| `-physical_icc2_blocks` *block_name_list* | list | For NDM format IC Compiler II designs, a list of block names, which are read in from the library directory path specified by the `-physical_icc2_lib` option. The tool reads only the physical data for the specified blocks. |
| `-physical_enable_clock_data` | none | Enables the use of physical data for clock networks. Without this option, clock nets are ignored. |

Feedback

| Option and Argument | Data Type | Description |
|---|---|---|
| `-physical_enable_all_vias` | none | Enables the use of physical data for all vias. |

### Description

The `set_layout_database_options` command specifies the design to be analyzed.

For LEF/DEF designs, you must use the `-physical_tech_lib_path`, `-physical_lib_path`, and `-physical_design_path` options.

For NDM designs, you must use the `-physical_icc2_lib` and `-physical_icc2_blocks` options. You can optionally use the `-physical_lib_path` option.

Some of these options take a list of files as an argument. The Tcl syntax provides several ways to express a list, including the following:

- Enclose the list within curly braces. For example:

  ```
  -physical_icc2_blocks {block1 block2 block3}
  ```

- Enclose the list within square brackets and include the string “list”. For example:

  ```
  -physical_icc2_blocks [list block1 block2 block3]
  ```

If a list contains only a single item, you can use the item without braces or brackets. The following examples are all acceptable:

```
-physical_icc2_blocks block1
-physical_icc2_blocks {block1}
-physical_icc2_blocks [list block1]
```

If a list is long, use the backslash (\) character to indicate continuation to the next line. For clarity, you might want to place each list item on a separate line; however, spaces are acceptable delimiters between list items. For example:

```
-physical_icc2_blocks {block1 \
                       block2 block3  \
                       block4}

-physical_icc2_blocks [list block1 \
                       block2 block3  \
                       block4]
```

Feedback

# set_power_ground_nets

Ignores the resistance of power and ground nets.

### Syntax

```
set_power_ground_nets
    [-nets net_name]
    [-case_sensitive YES | NO]
    [-disable]
    [-enable]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-nets` | string | Specifies the power and ground nets; wildcards are supported. |
| `-case-sensitivity` | string | Specifies if net names are case-sensitive during selection. Default: `YES` |
| `-disable` | n/a | Disables power and ground net flow. |
| `-enable` | n/a | Enables power and ground net flow. |

Feedback

# starrc_gpd_read_opens_shorts

Creates an error file that contains information in the vicinity of opens and shorts for specified nets or regions.

### Syntax

```
starrc_gpd_read_opens_shorts
     [-gpd gpd_dir]
     [-error_file err_file]
     [-window bbox]
     [-type err_type]
     [-limit limit]
     [-add_gui_selection]
     [-add_net_attributes option]
     [-nets net_name]
     [-summary_view]
     [-shorts_types err_type]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-gpd` *gpd_dir* | string | The GPD generated from the StarRC extraction. The argument is the GPD directory. |
| `-error_file` *err_file* | string | An optional file name for the opens and shorts information; the default is starrc_openshort.err. |
| `-window` *bbox* | list | The design region to load. If this option is not used, the entire database is loaded. The argument is a list {x1,y1,x2,y2}, where x1 and y1 define the lower-left corner of the region and x2 and y2 define the upper-right corner. |
| `-type` *err_type* | string | Specifies which errors to analyze. Valid values are `open`, `short`, and `all` (default). |
| `-limit` *limit* | integer | The maximum number of errors to display |
| `-add_gui_selection` | Boolean | If used, nets with opens and shorts are highlighted when the GUI is started. On by default. |
| `-add_net_attributes` *option* | string | Valid values are `replace` (default) and `append`. If `replace`, creates user attributes starrc_drc_violation and starrc_drc_coordinates. If `append`, previous attributes are not removed. |
| `-nets` *net_name* | string | Nets for which to save extra information around opens and shorts. Can be a single net or a space-delimited list of nets inside double quotation marks. Wildcard * is supported. |

Feedback

| Option and Argument | Data Type | Description |
|---|---|---|
| `-summary_view` | string | Each type of shorts and opens errors gets distinctive color of X marker to categorize the errors. See Managing Open and Short Errors Using Summary View. |
| `shorts_types` *err_type* | string | Specifies which short error types to analyze. Valid values are net unselectable, nonselected, skip_cell, fill, and blockage.<br>In the Parasitic Explorer error browser GUI, you can<br>• Load types of shorts errors selectively<br>• Apply types of shorts errors to view the shorts errors<br>• Load and sort the selected types of shorts errors for debugging |

### Description

The `starrc_gpd_read_opens_shorts` command creates an error file that contains detailed information in the vicinity of opens and shorts for specified nets or regions. The GPD and the star directory created after a StarRC extraction run are the sources of the design information. The Parasitic Explorer error browser then uses the generated error file to display layout information for the selected nets or regions.

### Examples

The following example lists only those number of opens errors that fit within the specified design region (bounding box) in the Error Browser GUI:

```
starrc_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -window
 0,0,10000,5000 -type open
```

The following example lists only those number of shorts errors that fit within the specified design region (bounding box) in the Error Browser GUI, where the shorts errors types are `net unselectable`, `nonselected`, `blockage`, and `power`:

```
starrc_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -window
 0,0,10000,5000 -type short  \
-short_types (net unselectable nonselected blockage power)
```

The following example lists all shorts errors of the blockage type in the summary view:

```
starrc_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -summary_view
 -type short  -short_types blockage
```

The following example lists only that number of shorts errors that fit within the specified design region (bounding box) in the summary view:

```
starrc_shell> starrc_gpd_read_opens_shorts -gpd top.gpd -summary_view
 -type short  -window 0,0,10000,5000
```

Feedback

If you want to view all shorts and opens errors from the actual extracted database, instead of limiting only to the list of specified nets, use the commands as shown in the following example::

```
% StarXtract star_cmd
% StarXtract -write_short_regions -nets_file nets_file star_cmd
starrc_shell> starrc_gpd_read_opens_shorts -gpd <gpd_directory>
 -error_file error_file.txt
```

Before invoking the StarRC shell, use the `StarXtract` command to access the original GPD directory and add more nets to see additional errors. Then, use the `starrc_gpd_read_opens_shorts` command to view open and short errors in the Error Browser GUI.

**See Also**

- Analyzing Open and Short Errors
- Managing Open and Short Errors Using Summary View
- Viewing and Analyzing Open and Short Errors

Feedback

# start_gui

Invokes the Parasitic Explorer graphical user interface. The `gui_start` command is equivalent to the `start_gui` command.

Feedback

# write_parasitics

Writes a GPD or DSPF file that is read by the Parasitic Explorer tool.

### Syntax

```
write_parasitics
     [-pe]
     [-format file_format]
```

### Arguments

| Option and Argument | Data Type | Description |
|---|---|---|
| `-pe` | | Allows the Parasitic Explorer tool to read the parasitics file |
| `-format` | string | Specifies the file format with parasitics information |

### Description

The `write_parasitics` command writes the GPD or DSPF file after RC scaling in the Parasitic Explorer tool.

```
starrc_shell> write_parasitics -pe -format gpd test.gpd

starrc_shell> write_parasitics -pe -format dspf test.spf
```

### See Also

- scale_parasitics

Feedback

# Other Supported Commands

Table 7 lists the supported commands. Many of these commands are used in common with other Synopsys tools. This list does not include all possible Tcl commands. For more information, see the command man pages.

*Table 7 Supported Commands*

| Supported Commands | |
|---|---|
| add_to_collection | parallel_foreach_in_collection |
| alias | parse_proc_arguments |
| all_inputs | post_eval |
| all_outputs | print_suppressed_messages |
| append_to_collection | printenv |
| apropos | printvar |
| complete_net_parasitics | proc_args |
| copy_collection | proc_body |
| cputime | query_objects |
| current_design | quit |
| current_instance | read_parasitics |
| create_pe_data | redirect |
| date | remove_annotated_parasitics |
| define_proc_attributes | remove_design |
| echo | remove_from_collection |
| error_info | remove_host_options |
| exit | remove_license |
| filter_collection | remove_license_limit |
| find | remove_user_attribute |
| foreach_in_collection | rename |
| get_app_var | report_annotated_parasitics |

Feedback

*Table 7 Supported Commands (Continued)*

| Supported Commands | |
|---|---|
| get_attribute | report_app_var |
| get_cells | report_attribute |
| get_command_option_values | report_hierarchy |
| get_defined_attributes | report_host_usage |
| get_defined_commands | report_license_limit |
| get_designs | report_units |
| get_license | set_app_var |
| get_message_ids | set_host_options |
| get_nets | set_license_limit |
| get_pins | set_units |
| get_ports | set_user_attribute |
| getenv | setenv |
| help | sh |
| history | sizeof_collection |
| index_collection | sort_collection |
| is_false | source |
| is_true | start_hosts |
| list_attributes | start_profile |
| list_designs | stop_hosts |
| list_key_bindings | stop_profile |
| list_licenses | suppress_message |
| lminus | unalias |
| ls | unsuppress_message |
| man | which |
| mem | write_app_var |

Feedback

*Table 7 Supported Commands (Continued)*

| Supported Commands | |
|---|---|
| parallel_execute | write_parasitics |