

# **CustomSim™ Command Reference**

---

Version L-2016.03, March 2016

**SYNOPSYS®**

# Copyright and Proprietary Information Notice

©2016 Synopsys, Inc. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

## 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 <http://www.synopsys.com/Company/Pages/Trademarks.aspx>.

All other product or company names may be trademarks of their respective owners.

## 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.

Synopsys, Inc.  
690 E. Middlefield Road  
Mountain View, CA 94043  
[www.synopsys.com](http://www.synopsys.com)

Printed in U.S.A.

# Contents

---

Related Publications .....	ix
Conventions .....	x
Customer Support .....	xi
<hr/>	
<b>1. Using CustomSim Commands and Options .....</b>	<b>1</b>
Using CustomSim Commands.....	1
Using a Command Script File .....	2
Using Commands Within the Netlist.....	2
Using Commands During Interactive Debugging Mode.....	2
Interactive mode with Tcl Commands .....	3
Enabling Tcl Mode .....	3
Common Syntax Definitions.....	4
Common 3D-IC Syntax Definitions .....	6
Command Categories .....	8
<hr/>	
<b>2. CustomSim Batch Command Syntax.....</b>	<b>11</b>
check_node_excess_rf .....	13
check_node_hotspot .....	17
check_node_quick_rf.....	18
check_node_zstate .....	21
check_timing_edge .....	24
check_timing_hold .....	29
check_timing_pulse_width.....	32
check_timing_setup .....	35
enable_print_statement .....	38
force_node_voltage .....	38
keep_top_element .....	40
load_ba_file .....	40

**Contents**

load_gndcap_file . . . . .	49
load_operating_point . . . . .	52
load_parameter_file . . . . .	54
load_vector_file . . . . .	55
load_verilog_file . . . . .	55
map_ba_terminal . . . . .	56
meas_post . . . . .	58
probe_waveform_current . . . . .	59
probe_waveform_ixba . . . . .	65
probe_waveform_logic . . . . .	70
probe_waveform_va . . . . .	72
probe_waveform_voltage . . . . .	74
pulse_oscillator . . . . .	81
release_node_voltage . . . . .	82
report_dangling_node . . . . .	83
report_floating_node . . . . .	83
report_model . . . . .	85
report_node_alias . . . . .	86
report_node_cap . . . . .	87
report_operating_point . . . . .	91
report_power . . . . .	93
report_sim_activity . . . . .	108
set_active_net_flow . . . . .	112
set_analysis_core . . . . .	113
set_array_option . . . . .	114
set_ba_option . . . . .	115
set_bus_format . . . . .	124
set_capacitor_option . . . . .	125
set_ccap_level . . . . .	127
set_ccap_option . . . . .	130
set_circuit_flash . . . . .	131

set_current_option . . . . .	133
set_dc_option . . . . .	133
set_duplicate_rule . . . . .	137
set_flash_option . . . . .	140
set_floating_node . . . . .	141
set_hotspot_option . . . . .	143
set_identification_rule . . . . .	144
set_inductor_option . . . . .	149
set_interactive_stop . . . . .	150
set_latch_control . . . . .	152
set_logic_threshold . . . . .	154
set_meas_dump . . . . .	155
set_meas_format . . . . .	156
set_message_option . . . . .	156
set_model_level . . . . .	159
set_model_option . . . . .	161
set_monte_carlo_option . . . . .	168
set_multi_core . . . . .	175
set_multi_rate_option . . . . .	177
set_oscillator . . . . .	177
set_parameter_value . . . . .	179
set_partition_option . . . . .	180
set_postlayout_meas . . . . .	181
set_powernet_level . . . . .	186
set_powernet_option . . . . .	187
set_probe_option . . . . .	188
set_probe_window . . . . .	190
set_ra_functional_resistor . . . . .	192
set_ra_net . . . . .	194
set_ra_net_type . . . . .	195
set_ra_option . . . . .	196

**Contents**

set_ra_pwnet_driver . . . . .	198
set_ra_pwnet_option . . . . .	199
set_ra_reuse . . . . .	200
set_rc_network_option. . . . .	202
set_resistor_option. . . . .	202
set_restore_option . . . . .	208
set_sample_point. . . . .	208
set_save_state. . . . .	209
set_sim_case. . . . .	212
set_sim_hierid . . . . .	213
set_sim_level . . . . .	214
set_sram_characterization. . . . .	215
set_synchronization_level . . . . .	218
set_synchronization_option . . . . .	218
set_tolerance_level . . . . .	219
set_tolerance_option . . . . .	221
set_va_view . . . . .	223
set_vector_char . . . . .	226
set_vector_option. . . . .	227
set_waveform_option. . . . .	230
set_waveform_sim_stat . . . . .	236
set_wildcard_rule. . . . .	238
set_zstate_option. . . . .	239
skip_circuit_block. . . . .	242
source . . . . .	243
<hr/>	
3. CustomSim Interactive Command Syntax. . . . .	245
CustomSim Interactive Commands . . . . .	246
alias . . . . .	248
icheck_node_zstate . . . . .	248
iclose_log . . . . .	251
icontinue_sim . . . . .	252

idelete_break_point . . . . .	253
iforce_node_voltage . . . . .	254
ilist_break_point . . . . .	254
ilist_force_node . . . . .	255
imatch_elem . . . . .	256
imatch_node . . . . .	256
iopen_log . . . . .	258
iprint_connectivity . . . . .	259
iprint_dcpath . . . . .	260
iprint_elem_info . . . . .	262
iprint_exi . . . . .	264
iprint_flash_cell . . . . .	266
iprint_help . . . . .	267
iprint_node_info . . . . .	268
iprint_subckt . . . . .	268
iprint_time . . . . .	269
iprint_tree . . . . .	269
iprobe_waveform_current . . . . .	270
iprobe_waveform_voltage . . . . .	272
iquit_sim . . . . .	275
irelease_node_voltage . . . . .	275
ireport_node_cap . . . . .	276
ireport_operating_point . . . . .	280
isearch_node . . . . .	281
iset_break_point . . . . .	284
iset_diagnostic_option . . . . .	284
iset_env . . . . .	285
iset_interactive_option . . . . .	286
iset_interactive_stop . . . . .	287
iset_save_state . . . . .	288
iset_zstate_option . . . . .	289
DC Interactive Mode Commands . . . . .	291
exit . . . . .	292
iclose_log . . . . .	293
icontinue_dc . . . . .	293
idelete_node_ic . . . . .	293
imatch_elem . . . . .	294
imatch_node . . . . .	295
iopen_log . . . . .	296

**Contents**

---

iprint_connectivity . . . . .	297
iprint_elem_info . . . . .	298
iprint_exi . . . . .	301
iprint_help . . . . .	302
iprint_node_info . . . . .	303
isearch_node . . . . .	304
iset_node_ic . . . . .	306
<hr/>	
<b>4. CustomSim Dynamic CircuitCheck . . . . .</b>	309
cck_excess_ipath. . . . .	309
cck_post. . . . .	312
cck_signal . . . . .	315
cck_soa . . . . .	324
cck_substrate. . . . .	336
cck_toggle_count. . . . .	341
cck_analog_pdown . . . . .	349
cck_resistive_path_report . . . . .	354
<hr/>	
<b>A. Converting the WDF Waveform Output File Format . . . . .</b>	357
Using the convert2out Utility . . . . .	357
convert2out . . . . .	357
<hr/>	
<b>Index . . . . .</b>	363

## About This Manual

---

This manual describes the Synopsys CustomSim™ batch and interactive commands and options.

This chapter describes:

- [Related Publications](#).
- [Conventions used in this manual](#).
- How to access [Customer Support](#).

---

## Related Publications

For additional information about the CustomSim tool, see

- The documentation installed with the CustomSim software.
- The CustomSim Release Notes, available on SolvNet (see [Accessing SolvNet on page xi](#))
- Documentation on the Web, which provides HTML and PDF documents and is available on SolvNet (see [Accessing SolvNet on page xi](#))

You might also want to refer to the documentation for the following related Synopsys (and third-party) products:

- HSPICE®
- Eldo®
- Spectre®

# Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Courier	Indicates command syntax.
<i>Italic</i>	Indicates a user-defined value, such as <i>object_name</i> .
Purple	<ul style="list-style-type: none"> <li>■ Within an example, indicates information of special interest.</li> <li>■ Within a command-syntax section, indicates a default value, such as:</li> </ul> <pre>include_enclosing = true   false</pre>
Bold	<ul style="list-style-type: none"> <li>■ Within syntax and examples, indicates user input—text you type verbatim.</li> <li>■ Indicates a graphical user interface (GUI) element that has an action associated with it.</li> </ul>
[ ]	<p>Denotes optional parameters, such as:  <code>write_file [-f filename]</code></p>
...	Indicates that parameters can be repeated as many times as necessary: <code>pin1 pin2 ... pinN</code>
	Indicates a choice among alternatives, such as <code>low   medium   high</code>
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
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 the C key.

---

Convention	Description
{}	Indicates a definition that can be repeated as many times as necessary. For example: [-node] {node_name node_name}

---

## Customer Support

Customer support is available through [SolvNet](#) online customer support and through contacting the [Synopsys Technical Support Center](#).

---

### Accessing SolvNet

SolvNet includes an electronic knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services, which include downloading software, viewing Documentation on the Web, and entering a call to the Support Center.

To access SolvNet:

1. Go to the SolvNet Web page at <http://solvnet.synopsys.com>.
2. If prompted, enter your user name and password. (If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.)

If you need help using SolvNet, click Help on the SolvNet menu bar.

---

### Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a call to your local support center from the Web by going to <http://solvnet.synopsys.com/EnterACall> (Synopsys user name and password required).
- Send an e-mail message to your local support center.
  - E-mail support\_center@synopsys.com from within North America.

### Customer Support

- Find other local support center e-mail addresses at [http://www.synopsys.com/support/support\\_ctr](http://www.synopsys.com/support/support_ctr).
- Telephone your local support center.
  - Call (800) 245-8005 from within the continental United States.
  - Call (650) 584-4200 from Canada.
  - Find other local support center telephone numbers at [http://www.synopsys.com/support/support\\_ctr](http://www.synopsys.com/support/support_ctr).

## Using CustomSim Commands and Options

---

*Describes how to use CustomSim commands and options.*

This chapter describes:

- [Using CustomSim Commands](#)
- [Common Syntax Definitions](#)
- [Command Categories](#)

---

## Using CustomSim Commands

CustomSim commands are used to control simulator options and features. You can enter the commands in any of the following ways:

- [Using a Command Script File](#)
- [Using Commands Within the Netlist](#)
- [Using Commands During Interactive Debugging Mode](#)

All commands and their arguments are case-sensitive—even those commands placed within HSPICE® and Eldo® netlists. Command arguments that refer to netlist identifiers are treated as case-insensitive in HSPICE and Eldo format netlists, and as case-sensitive in Spectre® format netlists. See the *CustomSim User Guide* for more information about working with [HSPICE](#), [Eldo](#), or [Spectre](#) netlists.

The CustomSim tool uses Tcl, which is a non-proprietary scripting language, as its command language. To facilitate simple and easy use, many of the Tcl features are disabled by default. No previous knowledge of Tcl is required to use CustomSim commands. You can also set the CustomSim tool to use a full Tcl interpreter, as described in [Enabling Tcl Mode](#).

**Chapter 1: Using CustomSim Commands and Options**

Using CustomSim Commands

---

## Using a Command Script File

To use a command script file, enter a `-c` switch on the command line when starting the CustomSim tool, as shown in the following example:

```
xa netlist.sp -c command_file
```

All commands and settings in the command file are processed as if the command were placed on the final line of the netlist. That is, the commands used in the command file override any CustomSim commands used in the netlist file.

---

## Using Commands Within the Netlist

Enter CustomSim commands directly into the netlist using the `.option` or `options` statement appropriate for that netlist.

HSPICE and Eldo syntax netlists use an `.option` command to insert the CustomSim tool-specific commands into a netlist. See the following syntax:

```
.opt [ion] XA_CMD="xa_command arg arg arg"
```

For example:

```
.option XA_CMD="set_sim_level -level 5"
```

Spectre format netlists use an `options` analysis statement. See the following syntax:

```
user_xa_commands options xa_cmd="xa_command arg arg arg"
```

For example:

```
myoptions options xa_cmd="set_sim_level -level 5"
```

---

## Using Commands During Interactive Debugging Mode

The CustomSim tool provides an interactive mode, in which you can enter commands at the `XA >` command prompt. The CustomSim tool interactive mode commands are listed in [Chapter 3, CustomSim Interactive Command Syntax](#).

To enter the CustomSim tool interactive mode, use the `-intr` command-line flag. This command-line flag is used for all supported netlist formats.

-intr [time[unit]]

time

The time at which the CustomSim tool enters interactive mode. If you do not specify a time, the CustomSim tool does not enter interactive mode until you enter Ctrl-C.

unit

The unit of measure for time. Default is second. There can be no space between the time and the unit, or the CustomSim tool terminates the simulation with an error message.

Ctrl-C

After entering the -intr flag, you can optionally press Ctrl-C to enter interactive mode.

**Important:** If you do not specify -intr on the command line, the CustomSim tool cannot enter interactive mode. If you enter Ctrl-C, the CustomSim tool prompts you to abort or continue the simulation.

## Interactive mode with Tcl Commands

The interactive mode defaults to the Tcl OFF mode. You can change to Tcl ON using the -tcl command-line flag. All Tcl syntax rules are applied in the Tcl ON mode.

Refer to the [Enabling Tcl Mode](#) section for the special characters in the Tcl on mode.

## Enabling Tcl Mode

When invoking the simulator, you can enable a full Tcl interpreter by using the -tcl switch on the command line (Tcl ON behavior). The Tcl mode cannot be toggled on-and-off during a session—see [Table 1](#).

*Table 1    Tcl On-and-Off Behavior*

Character	Tcl ON behavior	Tcl OFF behavior
\$	Variable substitution	Literal \$
[ ]	Command substitution	Literal [ ]

**Chapter 1: Using CustomSim Commands and Options**

## Common Syntax Definitions

*Table 1    Tcl On-and-Off Behavior*

Character	Tcl ON behavior	Tcl OFF behavior
\	Backslash substitution	Same as Tcl ON
" "	Argument grouping	Same as Tcl ON
{ }	Argument grouping	Same as Tcl ON
#	Comment if first character on line	Same as Tcl ON
;	End of command delimiter	Same as Tcl ON

**Common Syntax Definitions**

The standard syntax conventions are documented in the [Conventions](#) section of this manual. There are also some common syntax definitions shared between many CustomSim commands, described in this section.

Several commands use *enable\_value* as an argument. Possible values are:

*Table 2    Possible Values for the enable\_value Argument*

Positive	Negative
1	0
yes	no
true	false
on	off

Several commands use the *instance\_spec* argument. The *instance\_spec* argument can take one of three different forms as shown:

**Syntax**

```
-inst inst_name [...]
-subckt subckt_name [...]
-inst inst_name [...] -subckt subckt_name [...]
```

## Arguments

Argument	Description
<i>inst_name</i>	The name of the instance to which the command applies. A full hierarchical instance name can be supplied. Wild cards are allowed.
<i>subckt_name</i>	The name of the subcircuit to which the command applies. The command is applied to all instances of that subcircuit type.

## Description

Commands such as `set_sim_level` and `set_va_view` take *instance\_spec* as an argument. The *instance\_spec* argument specifies the instances upon which the command is applied. The `-inst` switch is used alone to provide a set of instance names. The `-subckt` switch is used alone to provide a set of subcircuit names. When both the `-inst` and `-subckt` switches are used, the command is applied to the named instances—only in the named subcircuit.

### Example

[Example 1](#) applies a command to instances `x1.x2` and `x1.x4`:

*Example 1*

```
-inst x1.x2 x1.x4
```

[Example 2](#) applies a command to all instances of the subcircuit `pll`:

*Example 2*

```
-subckt pll
```

[Example 3](#) applies a command to all MOSFET transistor instances in the subcircuit `chgpump`:

*Example 3*

```
-inst m* -subckt chgpump
```

[Example 4](#) applies a command to all instances `x1` and `x2` in subcircuits `sub1` and `sub2`:

*Example 4*

```
-inst X1 X2 -subckt sub1 sub2
```

**Chapter 1: Using CustomSim Commands and Options**

## Common Syntax Definitions

An equivalent `instance_spec` to that previously shown:

```
-inst X1 X2 -subckt sub1  
-inst X1 X2 -subckt sub2
```

---

## Common 3D-IC Syntax Definitions

Integrated circuits (ICs) of different functions and technologies can be packaged as part of a three-dimensional IC (3D-IC) that needs to be simulated as a single circuit. Previously, the 3D-IC netlist syntax and features supported parallel configuration in 3D-IC integration using an interposer. Now, 3D-IC hierarchical configuration is supported to allow for more flexibility in 3D-IC integration, including stacked ICs. This enhancement allows the migration of an existing single IC technology flow to a multiple-technology flow, incorporating different process nodes, without major changes in the netlisting process.

Existing CustomSim commands and the probing syntax of voltages and currents in a circuit can be specified using wildcard patterns for instances and nodes. As an extension to this capability, CustomSim commands and probing syntax now support IC module scope. This could be useful in circuit analyses of module instances that appear at different levels of the circuit hierarchy.

The syntax is shown in the next section.

### Syntax

```
-inst [ic_module_label::] instance_spec  
-node [ic_module_label::] node_spec
```

### Arguments

---

Argument	Description
<code>ic_module_label::</code>	Specifies the name that references an IC module that is part of a 3D-IC. The IC module label acts as additional condition to compliment any existing scope filter options. So it forces the instance and node matching scopes specified with <code>-inst</code> or <code>-node</code> to be limited within the IC module.  All CustomSim commands that can take an <code>-inst</code> or <code>-node</code> argument can use the <code>ic_module_label</code> argument.

---

<b>Argument</b>	<b>Description</b>
<i>instance_spec</i>	Specifies the instances upon which the command is applied. You can use wildcard ("*") characters in instance names.
<i>node_spec</i>	Specifies the nodes upon which the command is applied. You can use wildcard ("*") characters in node names.

---

For example:

```
set_model_level -level 5 -inst cismod::*
```

Sets the model level to 5 for all instances in the `cismod` module.

The same syntax enhancement that supports the IC module label reference also applies to probing in an HSPICE netlist:

### Syntax

```
.print V([ic_module_label::]node_pattern)
.print I([ic_module_label::]instance_pattern)
```

### Arguments

---

<b>Argument</b>	<b>Description</b>
<i>ic_module_label::</i>	Specifies the IC module scope reference.
<i>instance_spec</i>	Specifies the wildcard pattern for node matching.
<i>node_spec</i>	Specifies the wildcard pattern for instance matching.

---

For example:

```
.probe v(cismod::* )
```

This command probes all node voltages under the scope of the `cismod` IC module.

## Command Categories

[Table 3](#) categorizes the CustomSim batch commands. See for a list of the interactive commands, see [Chapter 3, CustomSim Interactive Command Syntax](#).

*Table 3    CustomSim Command Categories*

Command Category	Command
Diagnostics	<code>check_node_excess_rf</code> <code>check_node_hotspot</code> <code>check_node_quick_rf</code> <code>check_node_zstate</code> <code>check_timing_edge</code> <code>check_timing_hold</code> <code>check_timing_pulse_width</code> <code>check_timing_setup</code> <code>force_node_voltage</code> <code>probe_waveform_current</code> <code>probe_waveform_logic</code> <code>probe_waveform_va</code> <code>probe_waveform_voltage</code> <code>release_node_voltage</code> <code>report_power</code> <code>report_sim_activity</code> <code>set_hotspot_option</code> <code>set_sample_point</code> <code>set_zstate_option</code>
Netlist Control	<code>keep_top_element</code> <code>pulse_oscillator</code> <code>set_va_view</code> <code>set_capacitor_option</code> <code>set_duplicate_rule</code> <code>set_floating_node</code> <code>set_inductor_option</code> <code>set_logic_threshold</code> <code>set_resistor_option</code> <code>set_wildcard_rule</code> <code>skip_circuit_block</code>

*Table 3 CustomSim Command Categories*

Command Category	Command
Output Control	<code>enable_print_statement</code> <code>report_model</code> <code>report_operating_point</code> <code>set_meas_format</code> <code>set_message_option</code> <code>set_probe_option</code> <code>set_probe_window</code> <code>set_waveform_option</code>
Post-layout / Back-annotation	<code>load_ba_file</code> <code>map_ba_terminal</code> <code>set_ba_option</code>
Speed and Accuracy Control	<code>set_ccap_level</code> <code>set_dc_option</code> <code>set_flash_option</code> <code>set_model_level</code> <code>set_model_option</code> <code>set_oscillator</code> <code>set_partition_option</code> <code>set_powernet_option</code> <code>set_sample_point</code> <code>set_sim_level</code> <code>set_tolerance_level</code> <code>set_tolerance_option</code>
Vector Stimulus	<code>load_vector_file</code>
Reliability Analysis	<code>set_ra_functional_resistor</code> <code>set_ra_net</code> <code>set_ra_net_type</code> <code>set_ra_option</code> <code>set_ra_pwnet_driver</code> <code>set_ra_pwnet_option</code> <code>set_ra_reuse</code>

## **Feedback**

### **Chapter 1: Using CustomSim Commands and Options**

#### **Command Categories**

## CustomSim Batch Command Syntax

---

*Provides the syntax for all of the CustomSim batch commands.*

The CustomSim commands are:

```
check_node_excess_rf
check_node_hotspot
check_node_quick_rf
check_node_zstate
check_timing_edge
check_timing_hold
check_timing_pulse_width
check_timing_setup
enable_print_statement
force_node_voltage
keep_top_element
load_ba_file
load_gndcap_file
load_operating_point
load_parameter_file
load_vector_file
load_verilog_file
map_ba_terminal
meas_post
probe_waveform_current
probe_waveform_ixba
probe_waveform_logic
probe_waveform_va
probe_waveform_voltage
pulse_oscillator
release_node_voltage
report_dangling_node
```

```
report_floating_node
report_model
report_node_alias
report_node_cap
report_operating_point
report_power
report_sim_activity
set_active_net_flow
set_analysis_core
set_array_option
set_ba_option
set_bus_format
set_capacitor_option
set_ccap_level
set_ccap_option
set_circuit_flash
set_current_option
set_dc_option
set_duplicate_rule
set_flash_option
set_floating_node
set_hotspot_option
set_identification_rule
set_inductor_option
set_interactive_stop
set_latch_control
set_logic_threshold
set_meas_dump
set_meas_format
set_message_option
set_model_level
set_model_option
set_monte_carlo_option
set_multi_core
set_multi_rate_option
set_oscillator
set_parameter_value
set_partition_option
set_postlayout_meas
set_powernet_level
set_powernet_option
```

```
set_probe_option
set_probe_window
set_ra_functional_resistor
set_ra_net
set_ra_net_type
set_ra_option
set_ra_pwnet_driver
set_ra_pwnet_option
set_ra_reuse
set_rc_network_option
set_resistor_option
set_restore_option
set_sample_point
set_save_state
set_sim_case
set_sim_hierid
set_sim_level
set_sram_characterization
set_synchronization_level
set_synchronization_option
set_tolerance_level
set_tolerance_option
set_va_view
set_vector_char
set_vector_option
set_waveform_option
set_waveform_sim_stat
set_wildcard_rule
set_zstate_option
skip_circuit_block
source
```

---

## check\_node\_excess\_rf

Performs an excessive rise/fall timing check.

### Syntax

```
check_node_excess_rf -node node_name {node_name}
    -title title_name
    [-fanout value]
```

```

[-rtime rise_time]
[-ftime fall_time]
[-utime ustate_time]
[-loth logic_low_voltage]
[-hith logic_high_voltage]
[-twindow tstart [{tstop}] {tstart [tstop]})
[-subckt subckt_name]
[-error_file output_file_name]
[-limit level]

```

---

Argument	Description
<code>-node <i>node_name</i> {<i>node_name</i>}</code>	Defines the signal node name, which can be the node name of a single node or a node name with the asterisk (*) wildcard character that represents a group of node names. The behavior of asterisk (*) character is controlled by <a href="#">setWildcardRule on page 238</a> . You must specify this argument first.
<code>-title <i>title_name</i></code>	Defines the title name of this excess rise/fall time check. Errors that occur during this check are listed in the .errt file. The name of the excessive rise/fall time error starts with this title name. This name is useful when there are multiple excessive rise/fall time checks used for a single run.
<code>-fanout <i>value</i></code>	Defines the fanout of driver nodes. If fanout is set to 1 (one), only the driver nodes with fanouts are checked to avoid an unnecessary check on internal nodes within logic gates. If fanout is set to 0 (zero), both the internal node and the driver node are checked. For example: fanout=0: all nodes (the default). fanout=1: nodes that have direct connection to transistor gate. fanout=2: nodes that have direct connection to transistor bulk.
<code>-rtime <i>rise_time</i></code>	Defines the rise time of the signal as time duration t2–t1. t1 is the time when the rising signal voltage crosses the voltage level <i>logic_low_voltage</i> . t2 is the time when the same continuously rising signal voltage crosses the voltage level <i>logic_high_voltage</i> . The default value is 1 ns.

Argument	Description
<code>-ftime fall_time</code>	Defines the fall time of the signal as t4-t3. t3 is the time when the falling signal voltage crosses the voltage <code>logic_high_voltage</code> . t4 is the time when the same continuously falling signal voltage crosses the voltage <code>logic_low_voltage</code> . The default value is 1 ns.
<code>-utime ustate_time</code>	Defines the time period when the signal voltage is between <code>logic_low_voltage</code> and <code>logic_high_voltage</code> . The default is 1ns. Note that <code>-utime</code> checking is only performed when a transition is incomplete, which means it fails to go from low-to-high or high-to-low.
<code>-loth logic_low_voltage</code>	Defines the logic low voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-hith logic_high_voltage</code>	Defines the logic high voltage for the signal and ref node. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-twindow tstart tstop {tstart tstop}</code>	Performs the check within the time window defined by <code>tstart tstop {tstart tstop}</code> . The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified it is be assumed to be the end of the simulation. The final <code>tstop</code> can also be the end or <code>END</code> keyword.
<code>-subckt subcircuit_name</code>	Performs the operation on all instances of the specified subcircuit. A wildcard character is not allowed when you specify the using the <code>subcircuit_name</code> .
<code>-error_file output_file_name</code>	Specifies the output error file name for any excessive rise/fall time violations. If the output file name is not specified, the default output file use the output prefix name by <code>-o</code> follow by <code>.errt</code> .  For example, if <code>-o xa_default</code> , then the output error file is <code>xa_default.errt</code> . If <code>-error_file</code> is specified, the output file name is the user-specified <code>output_file_name</code> with the <code>.errt</code> extension.

Argument	Description
-limit level	Specifies the hierarchical depth to which the nodes are printed when the asterisk (*) characters are used as the first character of a node name or subcircuit name. If not specified, this setting defaults to 3.

### Description

When performing an excess rise/fall time check, if violations occur, they are stored in a new CustomSim output file, *output\_file\_name.err*. The output file name is the prefix name from the XA -o command line argument. If -o is not specified, it is the prefix of the input file name.

By default, the CustomSim tool uses 30%/70% as the logic threshold value based on the rail-to-rail voltage it detects on the signal. The logic threshold value can be modified by the [set\\_logic\\_threshold](#) command. If -l0th and -hith are specified, they take precedence over the default logic threshold value specified by the [set\\_logic\\_threshold](#) command if the same node name is applied to both commands.

### Examples

```
check_node_excess_rf -node output -title exrf_check1 -rtimes \
2ns -ftime 2ns
```

Sets up excessive rise/fall time check with output signal with title name exrf\_check1, and rise time of 2ns and fall time of 2ns.

```
check_node_excess_rf -node output -title exrf_check1 \
-rtimes 2ns -ftime 2ns -twindow 50ns 1000ns
```

Same as the previous example, but adds a start time of 50ns and stop time of 1000ns.

```
check_node_excess_rf -node output -title exrf_check1 \
-rtimes 2ns -ftime 2ns -utime 2ns -twindow 50ns 200ns 400ns \
800ns
```

Same as the previous example, but adds multiple start and stop time, start at 50ns stop at 200ns then start at 400ns and stop at 800ns. It also adds a u-state check of 2ns.

```
check_node_excess_rf -node dout* -title exrf_check1 \
-rtimes 2ns -ftime 2ns -error_file xa_excessive_rise_fall
```

Same as the first example, but adds a wildcard (\*) to match multiple signals and appends the output violation file to `xa_excessive_rise_fall.errt`.

---

## check\_node\_hotspot

Checks for hot spots in a design.

### Syntax

```
check_node_hotspot -node node_name {node_name}
```

Argument	Description
<code>-node node_name {node_name}</code>	Specifies the node names to check for hot spots.

### Description

This command checks for hot spots in a design and reports the information in an output file with suffix of `.hotspot`. Note that hot spot node analysis does not apply to input nodes.

The information for each specified node includes:

- Average node capacitance
  - Lumped from netlist, wiring, transistor gate, and diffusion capacitances.
- Toggle count
  - Total number of logic toggles for the node.
- Average charging current
  - Average current flowing into the capacitances connected to the node.
- Average discharge current
  - Average current flowing out of the capacitances connected to the node.

At the end of the output file, `check_node_hotspot` also reports:

- Total number of non-input nodes.
- Total number of nodes toggling.
- Total charging current.
- Total discharging current.

By default, nodes are not reported when the sum of the capacitive charging and discharging currents is less than the hot spot factor of 0.5 multiplied by the sum of the node with the largest capacitive currents. You can modify this suppression of small current nodes with the [set\\_hotspot\\_option](#) command.

---

## check\_node\_quick\_rf

Performs a rise/fall time check and reports rise/fall times less than the user-defined threshold.

### Syntax

```
check_node_quick_rf -node node_name {node_name}  
    -title title_name  
    [-fanout value]  
    [-rtime rise_time]  
    [-ftime fall_time]  
    [-loth logic_low_voltage]  
    [-hith logic_high_voltage]  
    [-twindow tstart [{tstop}] {tstart [tstop] }]  
    [-subckt subckt_name]  
    [-error_file file_name]  
    [-limit level]  
    [-except_node node_name {node_name}]
```

---

Argument	Description
<code>-node <i>node_name</i> {<i>node_name</i>}</code>	Defines the signal node name, which can be the node name of a single node or a node name with the asterisk (*) wildcard character that represents a group of node names. The behavior of asterisk (*) character is controlled by <a href="#">set_wildcard_rule on page 238</a> . You must specify this argument first.
<code>-title <i>title_name</i></code>	Defines the title name of this quick rise/fall time check. Errors that occur during this check are listed in the .errt file. The name of the quick rise/fall time error starts with this title name. This name is useful when there are multiple quick rise/fall time checks used for a single run.

Argument	Description
<code>-fanout value</code>	Defines the fanout of driver nodes. If fanout is set to 1 (one), only the driver nodes with fanouts are checked to avoid an unnecessary check on internal nodes within logic gates. If fanout is set to 0 (zero), both the internal node and the driver node are checked. For example: fanout=0: all nodes (the default). fanout=1: nodes that have direct connection to transistor gate. fanout=2: nodes that have direct connection to transistor bulk.
<code>-rtime rise_time</code>	Defines the rise time threshold of the signal as time duration $t_2 - t_1$ . $t_1$ is the time when the rising signal voltage crosses the voltage level ( <i>logic_low_voltage</i> ). $t_2$ is the time when the same continuously rising signal voltage crosses the voltage level ( <i>logic_high_voltage</i> ). If the calculated rise time less than this threshold, the command considers it a violation and reports it. The default value is 1 ns.
<code>-ftime fall_time</code>	Defines the fall time threshold of the signal as $t_4 - t_3$ . $t_3$ is the time when the falling signal voltage crosses the voltage ( <i>logic_high_voltage</i> ). $t_4$ is the time when the same continuously falling signal voltage crosses the voltage ( <i>logic_low_voltage</i> ). If the calculated fall time less than this threshold, the command considers it a violation and reports it. The default value is 1 ns.
<code>-loth logic_low_voltage</code>	Defines the logic low voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-hith logic_high_voltage</code>	Defines the logic high voltage for the signal and ref node. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-twindow tstart tstop {tstart tstop}</code>	Performs the check within the time window defined by $t_{start} t_{stop} \{t_{start} t_{stop}\}$ . The <i>tstart</i> and <i>tstop</i> must come in pairs, except for the final window where if <i>tstop</i> is not specified it is be assumed to be the end of the simulation. The final <i>tstop</i> can also be the end or END keyword.

Argument	Description
<code>-subckt subcircuit_name</code>	Performs the operation on all instances of the specified subcircuit. A wildcard character is not allowed when you specify the <code>subckt_name</code> .
<code>-error_file output_file_name</code>	Specifies the output error file name for any excessive rise/fall time violations. If the output file name is not specified, the default output file use the output prefix name by <code>-o</code> follow by <code>.errt</code> .  For example, if <code>-o xa_default</code> , then the output error file is <code>xa_default.errt</code> . If <code>-error_file</code> is specified, the output file name is the user-specified <code>output_file_name</code> with the <code>.errt</code> extension.
<code>-limit level</code>	Specifies the hierarchical depth to which the nodes are printed when the asterisk (*) characters are used as the first character of a node name or subcircuit name. If not specified, this setting defaults to 3.
<code>-except_node node_name {node_name}</code>	Specifies the nodes to be excluded from quick rise/fall time checking. You can use a wildcard character in the node name.

### Description

When performing an quick rise/fall time check, if violations occur, they are stored in a new CustomSim output file, `output_file_name.errt`. The output file name is the prefix name from the XA `-o` command line argument. If `-o` is not specified, the default input file name prefix is `xa`.

By default, the CustomSim tool uses 30%/70% as the logic threshold value based on the rail-to-rail voltage it detects on the signal. The logic threshold value can be modified by the [set\\_logic\\_threshold](#) command. If `-l0th` and `-hith` are specified, they take precedence over the default logic threshold value specified by the [set\\_logic\\_threshold](#) command if the same node name is applied to both commands.

### Examples

```
check_node_quick_rf -node output -title quick_rf_check1 -rtime \
2ns -ftime 2ns
```

Sets up quick rise/fall time check with `output` signal with title name `quick_rf_check1`, and rise time of `2ns` and fall time of `2ns`.

```
check_node_quick_rf -node output -title quick_rf_check1 \
-rtime 2ns -ftime 2ns -twindow 50ns 1000ns
```

Same as the previous example, but adds a start time of 50ns and stop time of 1000ns.

```
check_node_quick_rf -node dout* -title quick_rf_check1 \
-rtime 2ns -ftime 2ns -error_file xa_quick_rise_fall
```

Same as the first example, but adds a wildcard (\*) to match multiple signals and appends the output violation file to `xa_quick_rise_fall.errt`.

```
check_node_quick_rf -node output -title quick_rf_check1 -rtime \
2ns -ftime 2ns
```

Sets up a quick rise/fall time check with output signal with title name `quick_rf_check1` and reports a violation if the rise time is less than 2ns or fall time is less than 2ns.

---

## **check\_node\_zstate**

Checks for nodes in a high-impedance state.

### **Syntax**

```
check_node_zstate -node node_name {node_name}
                  -title title_name
                  [-ztime z_time]
                  [-twindow tstart [tstop] {tstart [tstop]}]
                  [-tstep tstep_value]
                  [-fanout <0|1|2>]
                  [-except_node node_name {node_name}]
                  [-subckt subckt_name {subckt_name}]
                  [-except_subckt subckt_name {subckt_name}]
                  [-report port|all] [-error_file output_file_name]
                  [-numv value]
```

Argument	Description
<code>-node node_name {node_name}</code>	Specifies the node names at which the high-impedance state is checked. The node name can be a single node or a node name with a wildcard character that represents a group of node names. The rule of asterisk (*) character is controlled by <a href="#">set_wildcard_rule</a> .
<code>-title title_name</code>	Specifies the title of the high-impedance check statement. All high-impedance states reported by this statement are printed under this title in the error report. This argument is useful when there are multiple check statements specified in a single run.
<code>-ztime z_time</code>	Defines the period for which a node must remain in a high impedance state to be reported as a high impedance node. If not specified, the default is 5ns.
<code>-twindow tstart tstop {tstart tstop}</code>	Performs the check within the time window defined by <code>tstart tstop {tstart tstop}</code> . The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified it is be assumed to be the end of the simulation. The final <code>tstop</code> can also be the <code>end</code> or <code>END</code> keyword.
<code>-tstep tstep_value</code>	Performs the high-impedance state check at discrete time points that are integer multiples of the specified <code>tstep_value</code> . The first time point is the <code>-tstart</code> value of the <code>-twindow</code> option. When you specify <code>-tstep</code> , the <code>-ztime</code> value is ignored.
<code>-fanout &lt;0 1 2&gt;</code>	Selects the type of nodes for high-impedance checking: <ul style="list-style-type: none"> <li>▪ A value of 0 enables all nodes.</li> <li>▪ A value of 1 enables nodes that have direct connection to a transistor gate.</li> <li>▪ A value of 2 enables all nodes that have direct connection to a transistor bulk.</li> </ul> If not specified, the default is 0.
<code>-except_node node_name {node_name}</code>	Specifies the nodes to be excluded from high-impedance checking. This argument should be used only when <code>-node node_name</code> contains wildcard characters.

Argument	Description
<code>-subckt subckt_name {subckt_name}</code>	Specifies the subcircuits to be checked. Wildcard characters are allowed in subcircuit names.
<code>-except_subckt subckt_name {subckt_name}</code>	Specifies the subcircuits to be excluded from high-impedance checking. This argument should be used only when <code>-node node_name</code> contains wildcard characters. Wildcard characters are allowed in subcircuit names.
<code>-report port all</code>	Specifies the type of reporting based on the subcircuit ports. The default is <code>all</code> to report global nodes and internal nodes. If you set it to <code>port</code> , all internal nodes are excluded and all global nodes are reported based on the subcircuit ports. If you specify this argument with <code>  -subckt</code> , it limits the reporting to only the specified subcircuits.
<code>-error_file output_file_name</code>	Specifies the file name to report high-impedance state. The file name is followed by <code>.errz</code> as the suffix. If not specified, the default prefix follows the output file name specified by <code>-o</code> command-line option.
<code>-numv value</code>	Limits the number of violations to be reported for each <code>check_node_zstate</code> command. It is applied locally. The default is to report all the violations.

### Description

This command enables the CustomSim tool to diagnose specified nodes staying in a high-impedance (floating) state for a specific period of time. The detected nodes are reported to an error file with suffix of `.errz`.

A node stays in a high-impedance state if there is no conducting path from any voltage source to the node. A conducting path consists of conducting elements. An element is conducting if that specific element meets the following criteria:

Device	Rule
NMOS	$V_{gs} > V_{th}$ (rule=1)    $I_{ds} > idsth$ (rule=2)    $V_g > VDD - 0.1$ (rule=3) The default for $idsth$ is $1e-8A$ . The default rule is 1 & 2.
PMOS	$V_{gs} < V_{th}$ (rule=1)    $I_{ds} > idsth$ (rule=2)    $V_g < 0.1$ (rule=3) The default for $idsth$ is $1e-8A$ . The default rule is 1 & 2.

Device	Rule
NPN	$V_{be} > V_{beth}$ The default for $V_{beth}$ is 0.64
PNP	$V_{be} < -V_{beth}$ The default for $V_{beth}$ is 0.64
Diode	$I(diode) > diode\_ith \parallel V(a,c) > diode\_vth$ The default for $diode\_ith$ is 1e-8A.
Resistor	Always conducting by default.
Inductor	Always conducting.
Capacitor	Never conducting.
Verilog-A	Never conducting.
Controlled sources	Always conducting.

---

## check\_timing\_edge

Performs an edge timing check.

### Syntax

```
check_timing_edge -node node_name {node_name}
    -title title_name -ref ref_name
    -min_time min_time
    -max_time max_time
    [-window window_limit]
    [-trigger trigger_type]
    [-loth logic_low_voltage]
    [-hiht logic_high_voltage]
    [-twindow tstart [tstop] {tstart [tstop]}]
    [-subckt subckt_name]
    [-error_file output_file_name]
    [-data_edge_type edge_type]
    [-ref_edge_type edge_type] [-rule 1|2]
```

Argument	Description
<code>-node node_name {node_name}</code>	Defines a signal node name which can be the node name of a single node or a node name with the asterisk (*) wildcard character that represents a group of node names. The wildcard is limited to one hierarchy only.
<code>-title title_name</code>	Defines the title name of this edge timing check. Errors that occur during this check are listed in the .errt file. The name of the edge timing error starts with this title name. This name is useful when there are multiple edge timing checks used for a single run.
<code>-ref ref_name</code>	Defines the reference node name. <code>ref_name</code> must be a single node name. A wildcard is not allowed.
<code>-min_time min_time</code>	Defines the lower boundary of the timing edge difference between the signal and reference nodes. A timing edge error is reported when the timing edge difference is less than <code>min_time</code> . The timing edge difference is calculated only for the pair of permissible state transitions at the signal node and the reference node.
<code>-max_time max_time</code>	Defines the upper boundary of the timing edge difference between the signal and reference nodes. A timing edge error is reported when the timing edge difference is greater than <code>max_time</code> . The timing edge difference is calculated only for the pair of permissible state transitions at the signal node and the reference node.
<code>-window window_limit</code>	Eliminates a timing edge error report if the timing edge difference exceeds <code>window_limit</code> .

Argument	Description
<code>-trigger trigger_type</code>	Sets the <i>trigger_type</i> to one of the following keywords: <code>ref</code> specifies that the edge checking is executed only when the reference edge occurs. Use this setting if the signal changes before the reference edge. <code>input</code> specifies that the edge checking is executed only when the input signal edge (i.e., data edge) occurs. Use this setting if the signal changes after the reference edge. <code>both</code> specifies that the edge checking is executed when either the reference edge or the input signal edge occurs. This is the default setting. Note this setting tends to be pessimistic and might report false edge timing errors. This is because the simulator reports errors right away when it detects and does not filter the errors with later events. Use <code>-window</code> to remove the false errors.
<code>-loth logic_low_voltage</code>	Defines the logic low voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-hith logic_high_voltage</code>	Defines the logic high voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-twindow tstart tstop {tstart tstop}</code>	Performs the check within the time window defined by <code>tstart tstop {tstart tstop}</code> , which means both the reference node time ( <code>Tref</code> ) and the checked node event time ( <code>Tsig</code> ) for a timing violation have to be within the time window. The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified is be assumed to be the end of the simulation. The final <code>tstop</code> can also be the end or <code>END</code> keyword.
<code>-subckt subcircuit_name</code>	Performs the operation on all instances of the specified subcircuit. When using the <code>-subckt subcircuit_name</code> argument, <code>node_name</code> and <code>ref_name</code> are the node names searched inside the subcircuit in which the node name must not contain a wildcard character.

Argument	Description
<code>-error_file output_file_name</code>	<p>Specifies the output error file name for any edge timing violations. If the output file name is not specified, the default output file use the output prefix name by <code>-o</code> follow by <code>.errt</code>.</p>
<code>-data_edge_type edge_type</code>	<p>For example, if <code>-o xa_default</code>, then the output error file is <code>xa_default.errt</code>. If <code>-error_file</code> is specified, the output file name is the user-specified <code>output_file_name</code> with the <code>.err</code> extension.</p>
<code>-ref_edge_type edge_type</code>	<p>Specify one of the following keywords for <code>edge_type</code>:</p> <ul style="list-style-type: none"> <li>▪ <code>rise</code> (default) to specify the rising edge of the target node.</li> <li>▪ <code>fall</code> to specify the falling edge of the target node.</li> <li>▪ <code>rf</code> to specify both the rising and falling edge of the target node.</li> <li>▪ <code>x</code> only has an effect when you specify <code>-rule 2</code>. When you specify the <code>x</code> keyword with <code>-rule 2</code>, the CustomSim tool only checks most the recent transition. It does not matter if it is falling or rising edge. If you specify <code>rf</code> with <code>-rule 2</code>, the CustomSim tool checks the most recent of both the rising and falling edge.</li> </ul> <p>Specify one of the following keywords for <code>edge_type</code>:</p> <ul style="list-style-type: none"> <li>▪ <code>rise</code> (default) to specify the rising edge of the reference node.</li> <li>▪ <code>fall</code> to specify the falling edge of the reference node.</li> <li>▪ <code>rf</code> to specify both the rising and falling edge of the reference node.</li> <li>▪ <code>x</code> only has an effect when you specify <code>-rule 2</code>. When you specify the <code>x</code> keyword with <code>-rule 2</code>, the CustomSim tool only checks most the recent transition. It does not matter if it is falling or rising edge. If you specify <code>rf</code> with <code>-rule 2</code>, the CustomSim tool checks the most recent of both the rising and falling edge.</li> </ul>

Argument	Description
-rule 1   2	<p>Selects the rule of edge timing checking. The default is 1, which checks at every transition. When set to 2, it follows the HSIM® behavior, which is to compare the edge time difference between the most recent target signal transition and reference signal transition only.</p> <p>The qualified target-reference signal pair can vary depending on their transition order. For example, if there is a target signal transition, this check looks for the reference signal transition, which happens prior to and also most recently to this target signal transition. Similarly, if there is a reference signal transition, this check looks for the target signal transition, which happens prior to and also most recently to this reference signal transition.</p>

### Description

When performing an edge timing check, if violations occur, they are stored in a new CustomSim output file, *output\_file\_name.err*. The output file name is the prefix name from the CustomSim –o command line argument. If –o is not specified, it is the prefix of the input file name.

By default, the CustomSim tool uses 30%/70% as the logic threshold value based on the rail-to-rail voltage it detects on the signal. The logic threshold value can be modified by the [set\\_logic\\_threshold](#) command. If –l<sub>oth</sub> and –h<sub>ith</sub> are specified, they take precedence over the default logic threshold value specified by the [set\\_logic\\_threshold](#) command if the same node name is applied to both commands.

### Examples

```
check_timing_edge -node data -ref ctrl -ref_edge_type rise \
-ttitle edge_check -min_time 2ns -max_time 5ns \
-error_file xa_edge
```

When node `data` has a state transition, such as at time t2, the time edge difference t2-t1 must be within the range of 2ns for the `-min_time` and 5ns for the `-max_time`. Otherwise, a timing edge error is reported to the output error file `xa_edge.errt` with the title name `edge_check`. Time t1 is the most recent rise state transition time at node ‘ctrl’ before time t2. When node `ctrl` has a rise state transition at time t4, the time edge difference t4-t3 must be within the range of 2ns and 5ns. Otherwise a timing edge error is also reported. The time t3 is the most recent state transition time at node `data` before time t4.

```
check_timing_edge -node data -ref ctrl -ref_edge_type rise \
-ttitle edge_check -min_time 2ns -max_time 5ns -trigger input \
-window 10ns -error_file xa_edge
```

Same as the previous example, except the edge error is reported only when t1-t2 is less than 2ns or is less than 10ns but greater than 5ns. The time t2 is the most recent state transition time at node data before time t1.

```
check_timing_edge -node input* \
-ref ctrl -ref_edge_type rise -title edge_check -min_time \
0ns -max_time 8ns
```

This example checks if the `input*` edge changes occurs between 0ns and 8ns of the `ctrl` rising edge, when the reference signal `ctrl` rises or `input*` changes. The error report to the default `output_name.errt` file with the title `edge_check`.

## **check\_timing\_hold**

Performs a hold timing check.

### Syntax

```
check_timing_hold -node node_name {node_name}
                  -ref ref_name
                  -hold_time hold_time
                  -title title_name
                  [-window window_limit]
                  [-hith logic_high_voltage]
                  [-loth logic_low_voltage]
                  [-twindow tstart [tstop] {tstart [tstop]}]
                  [-subckt subckt_name]
                  [-error_file output_file_name]
                  [-data_edge_type edge_type]
                  [-ref_edge_type edge_type]
```

Argument	Description
<code>-node node_name {node_name}</code>	Defines a signal node name which can be the node name of a single node or a node name with the asterisk (*) wildcard character that represents a group of node names. The wildcard is limited to one hierarchy only.

Argument	Description
<code>-ref ref_name</code>	Defines the reference node name and must be a single node name. A wildcard is not allowed.
<code>-hold_time hold_time</code>	Specifies the hold time. The signal should not change to the specified edge to check for rising or falling conditions (rising edge only, falling edge only, or both) between Tref (the time the reference signal edge changes) - <code>hold_time</code> until Tref if <code>hold_time</code> is positive. Note that the <code>-window window_limit</code> argument is ignored for a positive <code>hold_time</code> .
<code>-window window_limit</code>	Specifies a hold time window for negative <code>hold_time</code> so that any hold timing violation with a time difference ( $ Tref - <\text{signal change time}> $ ) bigger than <code>window_limit</code> is ignored.
<code>-loth logic_low_voltage</code>	Defines the logic low voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-hith logic_high_voltage</code>	Defines the logic high voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-loth logic_low_voltage</code>	Defines the logic low voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-twindow tstart tstop {tstart tstop}</code>	Performs the check within the time window defined by <code>tstart tstop</code> { <code>tstart tstop</code> }, which means both the reference node time (Tref) and the checked node event time (Tsig) for a timing violation have to be within the time window. The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified is be assumed to be the end of the simulation. The final <code>tstop</code> can also be the end or END keyword.
<code>-subckt subcircuit_name</code>	Performs the operation on all instances of the specified subcircuit. When using the <code>-subckt subcircuit_name</code> argument, <code>node_name</code> and <code>ref_name</code> are the node names searched inside the subcircuit in which the node name must not contain a wildcard character.

Argument	Description
<code>-error_file output_file_name</code>	Specifies the output error file name for any hold violations. If the output file name is not specified, the default output file use the output prefix name by –o followed by .errt. For example, if –o xa_default, then the output error file is xa_default.errt. If <code>-error_file</code> is specified, the output file name is the user-specified <code>output_file_name</code> with the .errt extension.
<code>-data_edge_type edge_type</code>	Specify one of the following keywords for <code>edge_type</code> : <ul style="list-style-type: none"><li>▪ <code>rise</code> (default) to specify the rising edge of the target node.</li><li>▪ <code>fall</code> to specify the falling edge of the target node.</li><li>▪ <code>rf</code> to specify both the rising and falling edge of the target node.</li></ul>
<code>-ref_edge_type edge_type</code>	Specify one of the following keywords for <code>edge_type</code> : <ul style="list-style-type: none"><li>▪ <code>rise</code> (default) to specify the rising edge of the reference node.</li><li>▪ <code>fall</code> to specify the falling edge of the reference node.</li><li>▪ <code>rf</code> to specify both the rising and falling edge of the reference node.</li></ul>

### Description

When performing a hold check, if violations occur, they are stored in a new CustomSim output file, `output_file_name.errt`. The output file name is the prefix name from the XA –o command line argument. If –o is not specified, it is the prefix of the input file name.

By default, the CustomSim tool uses 30%/70% as the logic threshold value based on the rail-to-rail voltage it detects on the signal. The logic threshold value can be modified by the [set\\_logic\\_threshold](#) command. If `-l0th` and `-hith` are specified, they take precedence over the default logic threshold value specified by the [set\\_logic\\_threshold](#) command if the same node name is applied to both commands.

### Examples

```
check_timing_hold -node data -title hold_check -ref clk \
-hold_time 1.2ns -error_file RAM_timing.error
```

Specifies a hold timing check with `clk` as the reference node and “data” as the data node. The title name is named `hold_check` and it uses the hold violation

time of 1.2ns. The command also specified the output error file to RAM\_timing.error, the actual output error file is RAM\_timing.error.errt.

```
check_timing_hold -node data -title \
hold_check - ref clk -hold_time 1.2ns -error_file \
RAM_timing.error
```

Specifies a hold timing check with clk as the reference node and data as the data node. The title name is named hold\_check and it uses the hold violation time of 1.2ns. The command also specified the output error file to RAM\_timing.error, the actual output error file is RAM\_timing.error.errt.

---

## **check\_timing\_pulse\_width**

Performs a pulse width timing check.

### **Syntax**

```
check_timing_pulse_width -node node_name {node_name}
  -title title_name
  -low_min_time low_min_time
  -low_max_time low_max_time
  -high_min_time high_min_time
  -high_max_time high_max_time
  [-loth logic_low_voltage]
  [-hihit logic_high_voltage]
  [-twindow tstart [tstop] {tstart [tstop]}]
  [-subckt subckt_name]
  [-error_file output_file_name]
```

---

<b>Argument</b>	<b>Description</b>
-node <i>node_name</i> { <i>node_name</i> }	Defines a signal node name which can be the node name of a single node or a node name with the asterisk (*) wildcard character that represents a group of node names.

Argument	Description
<code>-title title_name</code>	Defines the title name of this pulse width timing check. Errors that occur during this check are listed in the <code>.errt</code> file. The name of the pulse width timing error starts with this title name. This name is useful when there are multiple pulse width timing checks used for a single run.
<code>-low_min_time</code> <code>low_min_time</code>	Defines the minimum value of the low-state pulse. Each low state pulse (the time period the node stays at the logic 0 state) is required to be greater than <code>low_min_time</code> and less than <code>low_max_time</code> <ul style="list-style-type: none"> <li>. The minimum length of the low pulse is ignored (not checked) if <code>low_min_time</code> is negative.</li> </ul>
<code>-low_max_time</code> <code>low_max_time</code>	Defines the maximum value of the low-state pulse. Each low state pulse (the time period the node stays at the logic 0 state) is required to be greater than <code>low_min_time</code> and less than <code>low_max_time</code> . The maximum length of the low pulse is ignored (not checked) if is negative.
<code>-high_min_time</code> <code>high_min_time</code>	Defines the minimum value of the high-state pulse. Each high state pulse (the time period the node stays at the logic 0 state) is required to be greater than and less than <code>high_max_time</code> . The minimum length of the low pulse is ignored (not checked) if <code>high_min_time</code> is negative.
<code>-high_max_time</code> <code>high_max_time</code>	Defines the maximum value of the high-state pulse. Each high state pulse (the time period the node stays at the logic 0 state) is required to be greater than <code>high_min_time</code> and less than <code>high_max_time</code> . The maximum length of the low pulse is ignored (not checked) if <code>high_min_time</code> is negative.
<code>-loth</code> <code>logic_low_voltage</code>	Defines the logic low voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-hith</code> <code>logic_high_voltage</code>	Defines the logic high voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.

Argument	Description
<code>-twindow tstart tstop {tstart tstop}</code>	Performs the check within the time window defined by <code>tstart tstop {tstart tstop}</code> , which means both event times ( $t_1, t_2$ ) of a pulse's two edges for a pulse width violation have to be within the time window. The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified it is assumed to be the end of the simulation. The final <code>tstop</code> can also be the end or END keyword.
<code>-subckt subcircuit_name</code>	Performs the operation on all instances of the specified subcircuit. A wildcard character is not allowed when you specify the <code>subcircuit_name</code> .
<code>-error_file output_file_name</code>	Specifies the output error file name for any edge timing violations. If the output file name is not specified, the default output file use the output prefix name by -o followed by .errt.  For example, if -o xa_default, then the output error file is <code>xa_default.errt</code> . If <code>-error_file</code> is specified, the output file name is the user-specified <code>output_file_name</code> with the <code>.err</code> extension.

### Description

When performing an pulse width timing check, if violations occur, they are stored in a new CustomSim output file, `output_file_name.err`. The output file name is the prefix name from the `-o` command line argument. If `-o` is not specified, it is the prefix of the input file name.

By default, the CustomSim tool uses 30%/70% as the logic threshold value based on the rail-to-rail voltage it detects on the signal. The logic threshold value can be modified by the [set\\_logic\\_threshold](#) command. If `-l0th` and `-hith` are specified, they take precedence over the default logic threshold value specified by the [set\\_logic\\_threshold](#) command if the same node name is applied to both commands.

### Examples

```
check_timing_pulse_width -node out -title pulse_width_check \
-low_min_time 4ns -low_max_time 8ns -high_min_time 5ns \
-high_max_time 9ns
```

A pulse width violation error, following the title name of `pulse_width_check` is reported when the low pulse width at node `out` is less than `4ns` or greater

than 8ns; or when the high pulse width is less than 5ns or greater than 9ns. The default logic threshold value is used because -loth/-hith is not defined.

```
check_timing_pulse_width -node out -title pulse_width_check \
-low_min_time 4ns -low_max_time 8ns -high_min_time 5ns \
-high_max_time 9ns -subckt digital_ram
```

Same as the first example, but the out signal is applied to only node inside the subcircuit digital\_ram.

```
check_timing_pulse_width -node out -title pulse_width_check \
-low_min_time 4ns -low_max_time 8ns -high_min_time 5ns \
-high_max_time 9ns \ -error_file xa_pulse_width
```

Same as the first example, but output error file is append to xa\_pulse\_width.errt.

## check\_timing\_setup

Performs a setup timing check.

### Syntax

```
check_timing_setup -node node_name {node_name}
    -ref ref_name
    -setup_time setup_time
    -title title_name
    [-window window_limit]
    [-loth logic_low_voltage]
    [-hith logic_high_voltage]
    [-twindow tstart [tstop] {tstart [tstop]}]
    [-subckt subckt_name]
    [-error_file output_file_name]
    [-data_edge_type edge_type]
    [-ref_edge_type edge_type]
```

Argument	Description
-node node_name {node_name}	Defines a signal node name which can be the node name of a single node or a node name with the asterisk (*) wildcard character that represents a group of node names. The wildcard is limited to one hierarchy only.

Argument	Description
<code>-ref ref_name</code>	Defines the reference node name and must be a single node name. A wildcard is not allowed.
<code>-setup_time</code> <code>setup_time</code>	Specifies the setup time. The signal should not change to the specified edge to check for rising or falling conditions (rising edge only, falling edge only, or both) between Tref (the time the reference signal edge changes) - <code>setup_time</code> until Tref if <code>setup_time</code> is positive. Note that the <code>-window window_limit</code> argument is ignored for a positive <code>setup_time</code> .
<code>-window</code> <code>window_limit</code>	Specifies a setup time window for negative <code>setup_time</code> so that any setup timing violation with a time difference ( Tref - <signal change time> ) bigger than <code>window_limit</code> is ignored.
<code>-loth</code> <code>logic_low_voltage</code>	Defines the logic low voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-hith</code> <code>logic_high_voltage</code>	Defines the logic high voltage for the signal. The default is the same as the <a href="#">set_logic_threshold</a> value.
<code>-twindow tstart</code> <code>tstop {tstart</code> <code>tstop}</code>	Performs the check within the time window defined by <code>tstart tstop {tstart tstop}</code> , which means both the reference node time (Tref) and the checked node event time (Tsig) for a timing violation have to be within the time window. The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified is be assumed to be the end of the simulation. The final <code>tstop</code> can also be the end or END keyword.
<code>-subckt</code> <code>subcircuit_name</code>	Performs the operation on all instances of the specified subcircuit. When using the <code>-subckt subcircuit_name</code> argument, <code>node_name</code> and <code>ref_name</code> are the node names searched inside the subcircuit in which the node name must not contain a wildcard character.

Argument	Description
<code>-error_file output_file_name</code>	<p>Specifies the output error file name for any edge timing violations. If the output file name is not specified, the default output file use the output prefix name by <code>-o</code> followed by <code>.errt</code>.</p> <p>For example, if <code>-o xa_default</code>, then the output error file is <code>xa_default.errt</code>. If <code>-error_file</code> is specified, the output file name is the user-specified <code>output_file_name</code> with the <code>.errt</code> extension.</p>
<code>-data_edge_type edge_type</code>	<p>Specify one of the following keywords for <code>edge_type</code>:</p> <ul style="list-style-type: none"> <li>▪ <code>rise</code> (default) to specify the rising edge of the target node.</li> <li>▪ <code>fall</code> to specify the falling edge of the target node.</li> <li>▪ <code>rf</code> to specify both the rising and falling edge of the target node.</li> </ul>
<code>-ref_edge_type edge_type</code>	<p>Specify one of the following keywords for <code>edge_type</code>:</p> <ul style="list-style-type: none"> <li>▪ <code>rise</code> (default) to specify the rising edge of the reference node.</li> <li>▪ <code>fall</code> to specify the falling edge of the reference node.</li> <li>▪ <code>rf</code> to specify both the rising and falling edge of the reference node.</li> </ul>

### Description

When performing a setup check, if violations occur, they are stored in a new CustomSim output file, `output_file_name.err`. The output file name is the prefix name from the XA `-o` command line argument. If `-o` is not specified, it is the prefix of the input file name.

By default, the CustomSim tool uses 30%/70% as the logic threshold value based on the rail-to-rail voltage it detects on the signal. The logic threshold value can be modified by the [set\\_logic\\_threshold](#) command. If `-l0th` and `-hith` are specified, they take precedence over the default logic threshold value specified by the [set\\_logic\\_threshold](#) command if the same node name is applied to both commands.

### Examples

```
check_timing setup -node data -ref clk -title setup_check \
-setup_time 1.2ns
```

Specifies a setup timing check with `clk` as the reference node and `data` as the data node. The title name is named `setup_check` and it uses the setup violation time of `1.2ns`.

```
check_timing_setup -node data -data_edge_type fall -title \
setup_check1 -ref clk -ref_edge_type fall -setup_time 1.2ns
```

Specifies a setup timing check with `clk` as the reference node and `data` as the data node. Both the `clk` and `data` use the `fall` edge. The title name is `setup_check1` and it uses the setup violation time of `1.2ns`.

---

## **enable\_print\_statement**

Enables `.print` statements in HSPICE® and Eldo® netlists.

### **Syntax**

```
enable_print_statement -switch enable_value
```

### **Arguments**

See the [Common Syntax Definitions](#) section for details about the `enable_value` argument.

### **Description**

By default, the CustomSim tool treats all `.print` statements as if they were `.probe` statements. Use the `enable_print_statement` command to enable the ASCII output. The simulator generates a `.print` file containing the output from the `.print` statements.

### **Examples**

```
enable_print_statement yes
```

### **See Also**

[set\\_waveform\\_option](#)

---

## **force\_node\_voltage**

Forces the specified nodes to stay at the specified constant voltage.

## Syntax

```
force_node_voltage -node node_name {node_name}
    -voltage voltage_value
    [-time time_value]
    [-subckt subckt_name]
    [-slope t_value]
```

Argument	Description
-node <i>node_name</i> { <i>node_name</i> }	Specifies the node names to force.
-voltage <i>voltage_value</i>	Specifies the voltage value to set.
-time <i>time_value</i>	Specifies the time to force the nodes. The default is 0.
-subckt <i>subckt_name</i>	Forces only the nodes inside the specified subcircuit.
-slope <i>t_value</i>	Forces the voltage with a ramp of <i>t_value</i> (in seconds per volt). The default is 10p s/V. Note that this option is only applicable to <code>set_multi_rate_option -mode 2</code> .

## Description

`force_node_voltage` forces the specified nodes to stay at the specified constant voltage. The node voltage stays at the same value from the specified time until either the end of simulation or when the constant node voltage status is released by [release\\_node\\_voltage](#).

**Note:** Nodes of voltage sources, Verilog-A outputs, and nodes that have been optimized out cannot be forced/released.

## Examples

```
force_node_voltage vpump -voltage 2.5 -time 10ns
```

The `vpump` node is forced at 2.5V at 10 ns. This node remains at this value until the end of the simulation unless you use [release\\_node\\_voltage](#) for the same node.

## keep\_top\_element

Specifies the top-level instance to be simulated with respect to the other elements in the netlist.

### Syntax

```
keep_top_element -inst instance_name
```

Argument	Description
<code>-inst <i>instance_name</i></code>	Suppose the CustomSim tool uses a netlist, instances, and subcircuits with <code>keep_top_element</code> in the following sequence:  netlist A_nlist (instance A) (instance B) subckt (SA) subckt (SB)  With the command line:  <code>xa A_nlist -top SB -c cfg.xa</code> And <code>cfg.xa</code> contains:  <code>keep_top_element -inst A</code>  When you run the CustomSim tool, the top-level netlist has the A instance and all instances of the SB subcircuit.

### Description

This command works with the `-top` command line option to specify the instances and subcircuits in the top-level netlist. Except for the stimulus and voltage source elements, all of the elements in the original top-level netlist are ignored.

---

## load\_ba\_file

Specifies a postlayout back-annotation file.

## Syntax

```
load_ba_file [-file] filename
[-xba 0|1]
[-min_res value]
[-max_res value]
[-min_cap value]
[-min_ind value]
[-skipnet net_name {net_name}]
[-rcnet net_name {net_name}]
[-ccnet net_name {net_name}]
[-cnet net_name {net_name}]
[-ccap_to_gcap cap_value]
[-dpf switch_value]
[-add_netpin_by_xy_file file_name]
[-add_netpin DSPF_netname DSPF_node {DSPF_node}]
[-add_netpin_file netpin_file_name]
[-add_instpin_file file_name]
[-delete_netpin DSPF_netname DSPF_node {DSPF_node}]
[-delete_netpin_file netpin_file_name]
[-what_if cmd_file]
[-report_no_ba value]
[-layout_only_device_models model_name {model_name}]
```

Argument	Description
-file <i>filename</i>	Specifies the name of the parasitic netlist file (SPEF or SPF format). It can be a gzipped file to save the disk space. Star-RC™ can directly generate gzipped files. See the NETLIST_COMPRESS_COMMAND command in the <i>Star-RC Command Reference</i> .
-xba 0   1	Enables the extended back-annotation (XBA) flow. The default is 0. For more information about XBA, see the <a href="#">Enabling the XBA Flow section</a> .
-min_res <i>value</i>	Specifies the lower threshold of resistors from the specified back-annotation file to be kept. It overrides the value specified by <a href="#">set_ba_option</a> for the back-annotation file. All resistors with an absolute value below the specified value are shorted by <a href="#">set_resistor_option -rule=1</a> . The default is 0.01.

Argument	Description
<code>-max_res value</code>	Specifies the upper threshold of resistors from the specified back-annotation file to be kept. It overrides the value specified by <a href="#">set_ba_option</a> for the back-annotation file. All resistors with an absolute value above the specified value are treated as open-circuit by <code>set_resistor_option rule=1</code> . The default is <code>1e12</code> .
<code>-min_cap value</code>	Specifies the lower threshold of capacitors from the specified back-annotation file to be kept. It overrides the value specified by <a href="#">set_ba_option</a> for the back-annotation file. All capacitors with an absolute value below the specified value are treated as open-circuit by <code>set_capacitor_option rule=1</code> . The default is <code>1e-2*CSHUNT</code> . The default of CSHUNT is <code>1e-20</code> .
<code>-min_ind value</code>	Specifies the upper threshold of inductors from the specified back-annotation file to be kept. It overrides the value specified by <a href="#">set_ba_option</a> for the back-annotation file. All inductors above the specified value are shorted by <code>set_inductor_option rule=1</code> . The default is <code>0</code> .
<code>-skipnet net_name {net_name}</code>	Specifies the name of nets that should not be back- annotated. You can specify multiple net names. You can use wildcard characters in the net names.

Argument	Description
<code>-rcnet net_name {net_name}</code>	<p>Specifies the nets to use for full RC back-annotation. By default (when the <code>-rcnet</code> argument is not specified) all nets are back-annotated as full RC. If you specify <code>-rcnet</code>, then all other nets not specified by this argument only use lumped capacitance back-annotation. The lumped capacitance value uses the net capacitance value in the * NET line.</p> <p>This argument works in both SPF and SPEF formats and accepts wildcard characters.</p>
<code>-ccnet net_name {net_name}</code>	<p>Specifies for the nets to use Cg+Cc back-annotation (remove all resistors from a given net). All the nets not specified by this argument use full RC back-annotation (assuming the net itself contains RCs).</p> <p>This argument works in both SPF and SPEF formats and accepts wildcard characters.</p> <p>Also, you cannot use this argument with the <code>-lump_c_only</code> argument of the <a href="#">set_ba_option</a> command.</p>
<code>-cnet net_name {net_name}</code>	<p>Specifies for the nets to use lumped capacitance back-annotation. The lumped capacitance value uses the net capacitance value in the * NET line. All the nets not specified by this argument use full RC back-annotation (assuming the net itself contains RCs).</p> <p>This argument works in both SPF and SPEF formats and accepts wildcard characters.</p> <p>Also, you cannot use this argument with the <code>-lump_c_only</code> argument of the <a href="#">set_ba_option</a> command.</p>

Argument	Description
<code>-ccap_to_gcap cap_value</code>	<p>Converts all the coupling capacitors to ground capacitors if the value of the coupling cap is below the <i>cap_value</i>. If the back-annotation file contains no coupling capacitors, there is no impact.</p> <p>This option works for both SPF and SPEF format. The default is 1e-19.</p>
<code>-dpf switch_value</code>	<p>Specifies to back-annotate the DPF section. The default <i>switch_value</i> is 1.</p>
<code>-add_netpin_by_xy_file file_name</code>	<p>instructs the CustomSim tool to insert a "* P" net pin by specifying XY coordinates. It finds the closest location to the specified x/y coordinates on the defined metal layer. It uses the following formula to calculate the distance:</p> $d=\sqrt{(dX^2+dY^2)}$ <p>Where <math>dX = X_n - X_u</math>; <math>dY = Y_n - Y_u</math>  <math>X_n</math> and <math>Y_n</math> are node coordinates  <math>X_u</math> and <math>Y_u</math> - user-specified coordinates</p> <p>The specified file has the following file format.</p> <pre>net_name=&lt;net_name&gt; x=&lt;xcoord&gt; y=&lt;ycoord&gt; layer=&lt;layer_name&gt; [dist=&lt;xydist&gt;] [range=&lt;value&gt;]</pre> <p>This file format has the following rules.</p> <ul style="list-style-type: none"> <li>▪ <i>net_name</i>, <i>x</i>, <i>y</i>, and <i>layer</i> are required arguments. <i>x</i>, <i>y</i>, <i>dist</i>, and <i>range</i> are defined in DSPF file units.</li> <li>▪ <i>dist</i> and <i>range</i> are optional arguments. Use the default value '0' when the other arguments are not used.</li> <li>▪ You can specify multiple <i>net_name</i> entries inside this file. In the case of a duplicate <i>net_name</i>, the last one takes precedence.</li> <li>▪ Specify only one add netpin by xy coordinate command per line.</li> <li>▪ Wildcard characters are not supported.</li> <li>▪ Spaces are allowed between arguments and on either side of the equal sign (=).</li> </ul>

Argument	Description
<code>-add_netpin <i>DSPF_netname</i> <i>DSPF_node</i> {<i>DSPF_node</i>}</code>	Adds a new net pin. The first value is a DSPF net name, followed by one or more DSPF node names. The DSPF node name can be a sub-node, instance pin, or probe text node. You can specify multiple -add_netpin options.
<code>-add_netpin_file <i>netpin_file_name</i></code>	Specifies a user-defined file to add with two columns. The first column is a DSPF net name. The second column is a DSPF node name, with one DSPF net name - DSPF node name pair per line. The DSPF node name can be a sub-node, instance pin, or probe text node. You can specify only one -add_netpin_file option. Empty lines are supported. Lines start with '*' are ignored.
<code>-add_instpin_file <i>file_name</i></code>	Specifies a user-defined file with four columns per line. Each line defines the connection between the DSPF net to the prelayout device pin name in the following column format: <i>DSPF_netname</i> <i>DSPF_nodename</i> <i>Prelayout_instance_name</i> <i>Prelayout_port_name</i> <ol style="list-style-type: none"> <li>1. <i>DSPF_netname</i> is the net name from the DSPF file. For example: *   NET vdd ... where <i>vdd</i> is a DSPF net name.</li> <li>2. <i>DSPF_nodename</i> can be a net pin, instance pin, or sub-node.</li> <li>3. <i>Prelayout_instance_name</i> is a prelayout instance/device name.</li> <li>4. <i>Prelayout_port_name</i> is a prelayout port name for the instance/device</li> </ol> You can specify only one file name. Only the top-level DSPF is supported.

Argument	Description
<code>-delete_netpin <i>DSPF_netname</i> <i>DSPF_node</i> {<i>DSPF_node</i>}</code>	Deletes a net pin. The first value is a DSPF net name, followed by one or more DSPF node names. The DSPF node name can be a sub-node, instance pin, or probe text node. You can specify multiple -delete_netpin options.
<code>-delete_netpin_file <i>netpin_file_name</i></code>	Specifies a user-defined file to delete with two columns. The first column is a DSPF net name. The second column is a DSPF node name, with one DSPF net name - DSPF node name pair per line. The DSPF node name can be a sub-node, instance pin, or probe text node. You can specify only one -add_netpin_file option. Empty lines are supported. Lines start with '*' are ignored.

Argument	Description
<code>-what_if cmd_file</code>	<p>Specifies a command file to control what modifications can be done to the data read from the SPF file. You can apply only one command file to an SPF file. The command file supports the following commands:</p> <pre>modifyresistor net_name resistor_name new_value</pre> <p>This command specifies the new value for the resistor in the specified net. The width/area of the existing resistor is updated according to the ratio calculated as old resistance divided by new resistance.</p> <pre>addresistor net_name node1 node2 value [-layer layer_number   layer_name] [ -w   -a value]</pre> <p>This command specifies where the new resistor is to be added. If you do not specify a layer or width/area, the EM analysis is not performed for the resistor. The resistor is assigned a name as follows:</p> <pre>res_what_if_unique_number node1 node2</pre> <pre>addgcap net_name node1 value</pre> <p>This command specifies where the new ground capacitance is to be added.</p> <pre>modifyresistor vdd R3346 1000 addresistor vdd vdd:7 vdd:76 100 -layer metal1 -a 10 addgcap vdd vdd:193 1e-12</pre> <p>In this example, when applying the command-based what-if analysis feature, the CustomSim tool modifies the R3346 resistor that belongs to the vdd net to 1000 ohm. Also, a new resistor is added to the vdd net that connected between the vdd: 7 internal node and vdd: 76 internal node with the metal1 layer and area of 10 um*um. A new ground cap is added to the vdd net on the vdd:193 node with value of 1pF.</p>

Argument	Description
<code>-report_no_ba value</code>	Specify one of the following values: <ul style="list-style-type: none"><li>▪ 0 does not report missing SPF nets or instances (default).</li><li>▪ 1 reports instance names that contain missing SPF nets.</li><li>▪ 2 reports any missing SPF nets.</li></ul>
<code>-layout_only_device_models model_name {model_name}</code>	Provides the layout device model name that only exists in layout to allow proper back-annotation.

**Description**

Specifies a parasitic back-annotation file. You can specify multiple files using multiple `load_ba_file` commands. You can use a mixture of `spf` and `spef` formats.

***Enabling the XBA Flow***

The traditional back-annotation flow is based on device-to-device name matching between the prelayout and postlayout netlists. In some cases this approach does not provide 100% matching due to stacking device configurations, finger nets, and other discrepancies.

The XBA flow provides another solution by replacing the entire contents of prelayout subcircuit with the postlayout contents of the same subcircuit. It guarantees a clean (100% device and RC) back-annotation and incorporates existing RC back-annotation functionality. The assumption is that `SPF` file contains the `.SUBCKT` definitions. A top-level (flat) circuit definition is not supported.

After parsing of the prelayout netlist, preprocessing of the `SPF` file is done. During `SPF` parsing all the RC networks are collapsed and an ideal netlist is created from the instance section of the `SPF` file. Then the prelayout content of subcircuit to be back-annotated is replaced by ideal netlist.

After that the RC back-annotation flow runs. This approach provides correct back-annotation results and keeps all back-annotation features available, such as selective back-annotation. You can specify multiple `.SUBCKT` definitions in the `SPF` file.

During XBA, the following information is saved in the database:

- Node names specified by \* | NET statement in the SPF file ( / is replaced by .).
- Device names in the instance section of the SPF file.

You can only use those names in the analysis statement.

**Note:** The XBA flow has the following limitations.

- Subcircuit and instanced based simulation command control is not supported if the subcircuit or instance is nested inside the XBA subcircuit/instance.
- `probe_waveform_current [-isub|-x subckt_instance_name.port]` is not supported if the subcircuit or instance is nested inside the XBA subcircuit/instance.

## Examples

In the command script file:

```
load_ba_file -file parastic_net.spf
```

In an HSPICE or Eldo netlist:

```
.opt xa_cmd="load_ba_file -file parastic_net.spf -skipnet X1.in*"
```

In a Spectre® netlist:

```
baoptions options Xa_cmd="load_ba_file -file parastic_net.spf  
-skipnet X1.in*"
```

```
load_ba_file -file test.SPF - xba 1 -rcnet vdd
```

Enables the XBA flow and specifies net the VDD for full RC back-annotation.

## See Also

[map\\_ba\\_terminal](#), [check\\_node\\_excess\\_rf](#)

---

## load\_gndcap\_file

Lets you add ground capacitors to a node without modifying the netlist.

### Syntax

```
load_gndcap_file [-file] gcap_file {gcap_file}
```

<b>Argument</b>	<b>Description</b>
<i>gcap_file</i>	<p>Specifies the name of the capacitor file. This file contains the node names where the ground capacitors and values are added. It has the following format:</p> <pre>node_name1 cap_value1 [occ otc] node_name2 cap_value2 [occ otc] ... node_namen cap_valuen [occ otc]</pre> <p>You can specify anode name multiple times (with multiple entries), and the keyword of the first entry is used for all entries with the same node names.</p> <p>By default if no keyword is specified, the CustomSim tool adds a ground capacitor to the node. The <code>occ</code> keyword overwrites the constant capacitance. The <code>otc</code> keyword overwrites the total capacitance of the node.</p>

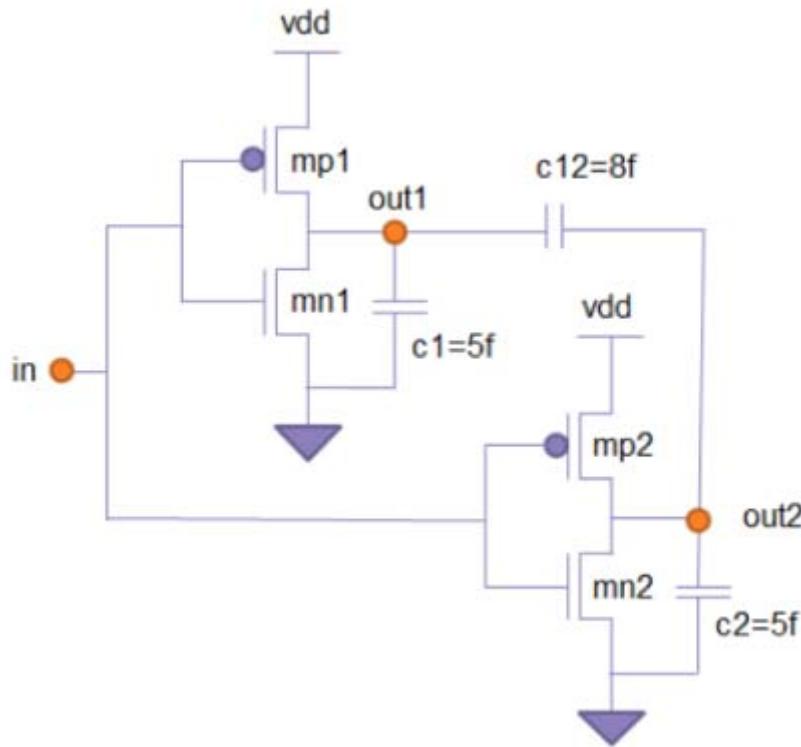
### **Description**

The format of the `load_gndcap_file` file is compatible with the `HSIMCAPFILE` format.

### **Examples**

The following examples are based on the schematic shown in [Figure 1](#) and assume that:

```
load_gndcap_file -file capfile
```



*Figure 1 Schematic Example for Adding Capacitance*

The CustomSim tool adds different values of capacitors to the specified nodes based on the content of the `capfile` file. If you do not specify a `capfile` file, the total capacitance for `out1` and `out2` is:

$$\text{totalc}(\text{out1}) = c(c1) + c(c12) + c(mp1+mn1) = 5\text{f} + 8\text{f} + 0.5605\text{f} = 13.5605\text{f}$$

$$\text{totalc}(\text{out2}) = c(c2) + c(c12) + c(mp2+mn2) = 5\text{f} + 8\text{f} + 0.5605\text{f} = 13.5605\text{f}$$

If `capfile` contains:

```
out1 35f
out2 10f
```

The CustomSim tool adds an additional 35f ground capacitor to `out1`, and 10f ground capacitor to `out2`:

$$\begin{aligned} \text{totalc}(\text{out1}) &= c(c1) + c(c12) + c(mp1+mn1) + c(\text{capfile:out1}) = 5\text{f} + 8\text{f} + 0.5605\text{f} \\ &+ 35\text{f} = 48.5605\text{f} \end{aligned}$$

**Chapter 2: CustomSim Batch Command Syntax**

`totalc(out2) = c(c2) + c(c12) + c(mp2+mn2) + c(capfile:out2) = 5f + 8f + 0.5605f  
+ 10f = 23.5605f`

If `capfile` contains:

```
out1 35f occ  
out2 10f occ
```

The CustomSim tool overwrites the constant capacitor of `out1` with 35f, and `out2` with 10f:

`totalc(out1) = c(capfile:out1) + c(c12) + c(mp1+mn1) = 35f + 8f + 0.5605f =  
43.5605f`

`totalc(out2) = c(capfile:out2) + c(c12) + c(mp2+mn2) = 10f + 8f + 0.5605f =  
18.5605f`

If `capfile` contains:

```
out1 35f otc  
out2 10f otc
```

The CustomSim tool overwrites the total capacitor of `out1` with 35f, and `out2` with 10f:

`totalc(out1) = c(capfile:out1) = 35f`

`totalc(out2) = c(capfile:out2) = 10f`

If `capfile` contains:

```
out1 35f otc  
out2 10f occ
```

Since `out1` is specified twice, the CustomSim tool only uses the keyword of the first entry (`otc`) to apply to all other nodes with the same node name. In this case, it overwrites the total capacitor of `out1` with 45f:

`totalc(out1) = c(capfile:out1) = 45f`

---

## **load\_operating\_point**

Determines how the CustomSim tool handles a file containing initial conditions.

## Syntax

```
load_operating_point [-file filename {filename}]
[-node_type node_type]
[-type ic_type]
[-subckt subckt {subckt}]
[-inst inst {inst}]
```

---

Argument	Description
-file <i>filename</i> { <i>filename</i> }	Specifies the name of the file to load.
-node_type <i>node_type</i>	Limits the application of initial conditions to nodes detected as <i>node_type</i> . A list of arguments is supported. Currently only the latch <i>node_type</i> is supported to limit the application of initial conditions to detected latches.
-type <i>ic_type</i>	Specifies the type of initial condition to be applied. The <i>ic_type</i> can be <i>ic</i> or <i>nodeset</i> . This option overrides what is specified in the file.
-subckt <i>subckt</i> { <i>subckt</i> }	Limits the application of initial conditions to nodes of the instances of <i>subckt</i> . A list of arguments is supported. Wildcards are not supported.
-inst <i>inst</i> { <i>inst</i> }	Limits the application of initial conditions to the specified instances. Wildcards are supported.

---

## Examples

```
load_operating_point -file op-file -node_type latch -type nodeset
```

Reads initial conditions from the *op-file* file and applies them as *nodeset* to latch nodes only.

```
load_operating_point -file op-file -inst xtop.analog
```

Reads initial conditions from the *op-file* file and applies them nodes in the *xtop.analog* instance only, as defined in the file (*.ic* or *.nodeset*).

```
load_operating_point -file op-file -subckt latch_clr -node_type
latch
```

Reads initial conditions from the `op-file` file and applies them only to latch nodes and to nodes in instances of the `latch_clr` subcircuit, as defined in the file (`.ic` or `.nodeset`).

---

## **load\_parameter\_file**

Lets you override instance parameter values in the netlist.

### **Syntax**

```
load_parameter_file -file filename
```

---

Argument	Description
<code>-file <i>filename</i></code>	Specifies the name of the file that contains the instance parameter information.

---

### **Description**

This command lets you specify a file that contains instance parameter values that override the corresponding definitions in the netlist. All other instance parameters keep their original values.

You can specify multiple lines in the file and use # to denote a comment. The file format for each line is:

```
instance_name.parameter_name value
```

You can use `load_parameter_file` in a `.alter` statement.

### **Examples**

```
load_parameter_file -file param.txt
```

The `param.txt` file contains:

```
x1.m1.dtemp -25
x1.m2.dtemp 0
x2.m1.dtemp 125
x2.m2.dtemp 25
```

In the previous example the `dtemp` parameter value overwrites the values for the `x1.m1`, `x1.m2`, `x2.m1`, and `x2.m2` instances. The new `dtemp` value is -25 for the `x1.m1` instance, 0 for the `x1.m2` instance, 125 for the `x2.m1` instance and 25 for the `x2.m2` instance. All other occurrences of `dtemp` keep their original values defined in the netlist.

## load\_vector\_file

Loads an HSPICE vector stimulus file or a VCD stimulus file. When VCD is the stimulus file format, the -ctl flag must be used to name the VCD control file.

### Syntax

```
load_vector_file -file filename  
[-format format_specification]
```

Argument-- <i>format_specification</i>	Description
VEC   vec	Specifies HSPICE vector file format.
VCD -ctl <i>filename</i>   vcd -ctl <i>filename</i>	Specifies a value change dump file format. The -ctl flag and <i>filename</i> are required.
EVCD [-ctl <i>filename</i> ]   evcd [-ctl <i>filename</i> ]	You can also specify an extended value change dump file format, as well as signal control file. If you do not specify the signal control file, the CustomSim tool follows the EVCD port direction rule and the unknown direction is ignored.

### Examples

#### *Example 5*

```
#load HSPICE vector stimulus  
load_vector_file -file input.vec -format VEC
```

#### *Example 6*

```
#load VCD stimulus  
load_vector_file -file stimulus.vcd -format VCD -ctl mapfile
```

## load\_verilog\_file

Loads a structural Verilog netlist. The CustomSim tool uses the connectivity from Verilog netlist but does not support Verilog functions.

### Syntax

```
load_verilog_file -file filename
```

Argument	Description
<code>-file filename</code>	Specifies the name of the structural Verilog netlist file.

**Examples**

```
#load a structural Verilog netlist
load_verilog_file -file top.v

#load multiple structural Verilog netlists
load_verilog_file -file inv.v
load_verilog_file -file ring_oscilator.v
```

---

## map\_ba\_terminal

Specifies the terminal name mapping between the back-annotation file and the terminal names recognized by the simulator.

**Syntax**

```
map_ba_terminal -name ba_file_term_name
    [-alias] valid_terminal_name
    [-subckt subcircuit_name]
```

**Arguments**

The `ba_file_term_name`, `valid_terminal_name`, and `subcircuit_name` arguments are user-defined.

**Description**

The CustomSim tool uses the first character, and optional subsequent characters, to determine which terminal is represented. [Table 4](#) shows the terminal identification characters.

If the instance terminals in the back-annotation file contains names that are not recognized based upon the characters shown in [Table 4](#), the `map_ba_terminal` command must be used.

*Table 4 map\_ba\_terminal Identification Characters*

Terminal Index	M (MOS)	Q (BJT)	R, C, D (Resistor, Capacitor, Diode)
1	D [R] [A] [I] [N]	C [O] [L] [L] [E] [C] [T] [O] [R]	A [N] [O] [D] [E] , P [L] [U] [S] , P [O] [S] [I] [T] [I] [V] [E]
2	G [A] [T] [E]	B [A] [S] [E]	B , C [A] [T] [H] [O] [D] [E] , M [I] [N] [U] [S] , N [E] [G] [A] [T] [I] [V] [E]
3	S [O] [U] [R] [C] [E]	E [M] [I] [T] [T] [E] [R]	S [U] [B] [S] [T] [R] [A] [T] [E]
4	B [U] [L] [K]	S [U] [B] [S] [T] [R] [A] [T] [E]	N/A

## Examples

In [Example 7](#), UDRN is used for the drain connection in the back-annotation file. The CustomSim tool does not recognize UDRN, so the following command must be used to back-annotate correctly:

```
map_ba_terminal -name UDRN -alias D
```

**Example 7 map\_ba\_terminal Back-annotation File Sample**

```
* |NET Xarr1/Xarr<11>/XI1/mc_n5 0.000826547PF
* |I (Xarr1/Xarr<11>/XI1/MM8:UDRN Xarr1/Xarr<11>/XI1/MM8 UDRN B 0 33.06
40.34)
* |I (Xarr1/Xarr<11>/XI1/MM5:SRC Xarr1/Xarr<11>/XI1/MM5 SRC B 0 32.85
40.815)
* |I (Xarr1/Xarr<11>/XI1/MM4:SRC Xarr1/Xarr<11>/XI1/MM4 SRC B 0 32.67
39.51)
* |I (Xarr1/Xarr<11>/XI1/MM3:UDRN Xarr1/Xarr<11>/XI1/MM3 UDRN B 0 32.95
39.035)
Cg39 Xarr1/Xarr<11>/XI1/MM8:UDRN 0 8.26142e-17
Cg40 Xarr1/Xarr<11>/XI1/MM5:SRC 0 1.2562e-17
Cg41 Xarr1/Xarr<11>/XI1/MM4:SRC 0 1.40334e-16
Cg42 Xarr1/Xarr<11>/XI1/MM3:UDRN 0 6.31581e-18
R51 Xarr1/Xarr<11>/XI1/MM8:UDRN Xarr1/Xarr<11>/XI1/MM5:SRC 10.3526
R52 Xarr1/Xarr<11>/XI1/MM8:UDRN Xarr1/Xarr<11>/XI1/MM4:SRC 31.6749
R53 Xarr1/Xarr<11>/XI1/MM4:SRC Xarr1/Xarr<11>/XI1/MM3:UDRN 3.52783
```

**See Also**[load\\_ba\\_file](#), [check\\_node\\_excess\\_rf](#)

---

## meas\_post

Performs measurements using existing simulation results.

**Syntax**

```
meas_post -waveform file_name
```

---

Argument	Description
-waveform <i>file_name</i>	You must specify an existing waveform file in fsdb or wdf format. Split waveforms are not supported.

---

**Description**

When you use `meas_post` in a command script file or with `.option xa_cmd`, the CustomSim tool:

- Only reads in the `.measure` commands with the related parameters in the netlist (when the `.measure` commands uses parameters).
- Performs the measurement specified in the netlist using the data in the specified waveform file.

You can use only one `meas_post` in a simulation. If you specify more than one command, the CustomSim tool uses the last one and ignores the previous one. the CustomSim tool issues a warning in the log file to point out which `meas_post` command was used and which ones were ignored.

When you run the CustomSim tool with `meas_post`, use the `-o` command line option to redirect the new output data. This option avoids overwriting the simulation log file.

You can use `meas_post` with a SPICE netlist that only includes `.measure` commands. This convention does not support wildcard characters.

### Examples

Suppose a previous CustomSim simulation generated the following files: `top.fsdb` and `top.log`. To perform a measurement for the simulation results, do the following steps:

1. Use a measurement command from a previous simulation or add a new one. For example:

```
.meas tran delay trig v(clk) val=1.5 rise=1 targ v(d[1])
val=1.5 rise=1
```

2. Add the following command in the existing command script file, for example:

```
meas_post -waveform top.fsdb
```

3. Run the CustomSim tool:

```
xa original_netlist -c config -o measNewResults
```

Two new files are created: `measNewResults.log` and `measNewResults.meas`.

## **probe\_waveform\_current**

Creates current waveform output.

### Syntax

```
probe_waveform_current -i | i1 instance_name {instance_name}
[-in instance_name {instance_name}]
[-iall instance_name {instance_name}]
[-isub | -x subckt_instance_name.port
{subckt_instance_name.port}]
[-subckt subckt_name]
[-limit level]
```

```

[-level level_val]
[-except_inst instance_name {instance_name}]
[-except_port except_pattern]
[-filetag file_tag]

```

Argument	Description
-il or -i <i>instance_name</i> { <i>instance_name</i> }	Specifies the name of the instance at which the current through instance terminal 1 is probed.
-in <i>instance_name</i>	Specifies the name of the instance at which the current of terminal <i>n</i> is probed, where <i>n</i> is a positive integer. Note this argument applies only to instances not specified by -subckt.
-iall <i>instance_name</i>	Specifies the name of the instance at which the current of all terminals are probed. Note this argument applies only to instances not specified by -subckt.
-isub   -x <i>subckt_instance_name.port</i> { <i>subckt_instance_name.port</i> }	-isub or -x probe returns the total current flowing into the subcircuit port. The <i>subckt_instance</i> is a subcircuit instance name, and <i>port</i> is a name of the subcircuit port. Positive values indicate the current is flowing into the subcircuit port. Negative values indicate the current is flowing out of the subcircuit port. Note that -x is not supported in Eldo netlist syntax.
-subckt <i>subckt_name</i>	Specifies the subcircuit of the instances at which the current is probed.
-limit <i>limit_val</i>	Specifies the hierarchy level down to which currents are probed when you use wildcard characters. A value of 0 specifies the top level. The default for <i>limit_val</i> is 3. When you specify -subckt, <i>limit_val</i> is relative to the scope of <i>subckt_name</i> . See <a href="#">Table 5</a> for probing behavior with different combinations of -limit and -level.

Argument	Description
<code>-level level_val</code>	Specifies the relative hierarchical level down to which currents are probed when you use wildcard characters. Numbering starts with 1 at the hierarchical level where the first wildcard appears. When you specify <code>-subckt</code> , <code>level_val</code> is relative to the scope of <code>subckt_name</code> in addition to the position of the first wildcard. See <a href="#">Table 5</a> for probing behavior with different combinations of <code>-limit</code> and <code>-level</code> .
<code>-except_inst instance_name {instance_name}</code>	Specifies the instances to exclude from probing. Note that this option applies only to instances.
<code>-except_port except_pattern</code>	Excludes ports that match the <code>except_pattern</code> from probing. You can specify wildcards in the pattern name. This option only works together with the <code>-isub</code> or <code>-x</code> option.
<code>-filetag file_tag</code>	Specifies a file tag to direct the signals probed with this command to a separate waveform file. The waveform file has the standard name with <code>.filetag</code> inserted before the normal file suffix, for example, <code>xa.filetag.wdf</code> .

### Description

Probes the current through an instance pin. The current waveform is written to the output file in the format specified by the `post` option in the netlist.

The CustomSim tool-supported wildcard characters ( \* ) can be used in the `instance_name` identifiers. The wildcard character can be used to match one level—or all levels—of the hierarchy. See the [setWildcardRule](#) command.

**Note:** The `probe_waveform_current` command cannot be used to probe subcircuit terminal currents.

Probes specified by this command are in addition to the `.probe` statement in the HSPICE or Eldo netlist files or `save` statements in the Spectre netlist files.

Long simulations, or simulations in which some nodes have a high level of activity, can produce very large waveform files. To minimize waveform file loading time in these files, you can direct signals to separate waveform files and keep file sizes smaller.

To direct signals to a separate waveform file, use the `-filetag` argument. All of the signals probed by that instance of the command are directed to a separate waveform file. The file name has the same format as the standard waveform file name, except for the suffix; for example, `xa.mytag.wdf`. All of the settings made by `set_waveform_option` also apply to the tagged output files. A signal can be directed to multiple files if it is probed with another command that specifies a different `file_tag`. Additionally, multiple `probe_waveform_current` commands can use the same file tag. The other probing commands (voltages and logic) can also use the same file tag.

When you use the `probe_waveform_current -isub|x subckt.port -except_port port_list`, the command returns the current flowing in the specified ports of the subcircuit specified after `-isub` or `-x` and excludes the current in the ports specified with `-except_port`.

The `-except_port` option can only work together with `-i` or `-x`. If `-except_port` is used, but neither `-isub` nor `-x` is specified in the `probe_waveform_current` command, then `-except_port` is ignored.

The `-except_port` option can work together with the following `probe_waveform_current` options:

- `-isub|x` (mandatory)
- `-subckt`
- `-except_inst`
- `-filetag`
- `-limit`

### Examples

```
probe_waveform_current -i vdd vss vda  
probe_waveform_current -iall X1.X2.M37  
probe_waveform_current * -except_inst *clk
```

The last example probes all instances down to the default level of hierarchy (3), except those finishing with the `clk` pattern.

```
probe_waveform_current mrxdrv -subckt rxblock -filetag rx_block
```

The previous example probes the `i1` current in `mrxdrv` in all instances of `rxblock` and diverts them to the `rx_block` waveform file. If the CustomSim tool is run as `xa net.sp -o xa -wavefmt wdf`, the waveform file is named `xa.rx_block.wdf`.

If the netlist contains:

```
.subckt inv in out
mn1 out in gnd gnd n w=5.00u l=1.00u
mp1 out in vdd vdd p w=10.00u l=1.00u
.ends
.subckt xfer gn gp d s
mp1 d gp s vdd p w=10.00u l=1.00u
mn1 d gn s gnd n w=5.00u l=1.00u
.ends
.subckt xor2 in1 in2 out
mn1 out n3 n2 gnd n w=5.00u l=1.00u
mp1 out n3 n4 vdd p w=10.00u l=1.00u
x2 in2 n3 inv
x3 n4 n2 inv
x4 n2 n4 n3 out xfer
x1 in1 n4 inv
.ends
x1 b a p xor2
x2 cin p sum xor2
```

And the configuration file contains:

```
probe_waveform_current -isub x1.* -except_port out
```

The `probe_waveform_current` command probes the currents flowing in all ports of instance `x1` except for the `out` port. You can see in the waveform file:

```
isub(x1.in1)
isub(x1.in2)

isub(x1.x1.in)
isub(x1.x2.in)
isub(x1.x3.in)
isub(x1.x4.gn)
isub(x1.x4(gp))
isub(x1.x4.d)
isub(x1.x4.s)
```

Using the same netlist as the previous example, but with the following `probe_waveform_current` command:

```
probe_waveform_current -isub * -subckt xor2 -except_port in*
```

This `probe_waveform_current` example probes all currents flowing in the XOR2 subcircuit ports and in the ports of the subcircuits below XOR2 in the hierarchy, except for the ports starting with `in`. You can see in the waveform file:

```
isub(x1.out)
  isub(x1.x1.out)
  isub(x1.x2.out)
  isub(x1.x3.out)
  isub(x1.x4.gn)
  isub(x1.x4.gp)
  isub(x1.x4.d)
  isub(x1.x4.s)
isub(x2.out)
  isub(x2.x1.out)
  isub(x2.x2.out)
  isub(x2.x3.out)
  isub(x2.x4.gn)
  isub(x2.x4.gp)
  isub(x2.x4.d)
  isub(x2.x4.s)
```

Using the same netlist as the first example, but with the following `probe_waveform_current` command:

```
probe_waveform_current -isub * -subckt xor2 -except_port in* -except_inst x2
```

This `probe_waveform_current` command probes all currents flowing in the XOR2 subcircuit ports, except for the ports starting with `in`. It also excludes the `x2` instance inside the XOR2 subcircuit. You can see in the waveform file:

```
isub(x1.out)
  isub(x1.x1.out)
  isub(x1.x3.out)
  isub(x1.x4.gn)
  isub(x1.x4.gp)
  isub(x1.x4.d)
  isub(x1.x4.s)
isub(x2.out)
  isub(x2.x1.out)
  isub(x2.x3.out)
  isub(x2.x4.gn)
  isub(x2.x4.gp)
  isub(x2.x4.d)
  isub(x2.x4.s)
```

The following example probes all currents from hierarchy level 1 to level 3 inside subcircuit volgen.

```
probe_waveform_current x1.* -level 3 -subckt volgen
```

### See Also

[probe\\_waveform\\_logic](#), [probe\\_waveform\\_va](#), [probe\\_waveform\\_voltage](#), [set\\_wildcard\\_rule](#)

## probe\_waveform\_ixba

Probes the sum of the all the device terminal currents of the specified sub-circuit instance touching the specified node. In the waveform file, the signal of the command has the syntax of *inst\_name\_to\_spf .i(\*:node)*.

### Syntax

```
probe_waveform_ixba -ipattern ipattern -node node_name
```

Argument	Description
-ipattern <i>ipattern</i>	Specifies the device pattern, or full hierarchical instance name down to the instance in the SPF file. You can use wildcard characters in the instance name. If the instance is not inside SPF file, the CustomSim tool issues a warning in the log file.
-node <i>node_name</i>	Specifies a top-level schematic node name to be traced. You cannot use wildcard characters in the node name.

### Description

The CustomSim tool supports extended back-annotation (XBA) flow (see the "Enabling the XBA Flow" section) that removes the contents of the specified prelayout subcircuit and replaces it with the contents of the Instance Section in the SPF file. The XBA flow ensures a clean back-annotation. However, because it creates a flat SPF netlist, the circuit hierarchy of the replaced subcircuit no longer exists, so the XBA flow has the following limitations.

- Subcircuit-based simulation command control is not supported.
- ISUB/X probing option is not supported.

- The `report_power` command is not supported.
- Prelayout instance names are replaced by postlayout instance names, so some commands need to change to reflect the instance name changes, such as probing.

This command addresses the ISUB/X probing limitation.

This command also supports using IXBA in the `.MEASURE` and `.PROBE` statements. The syntax is:

```
ixba(ipattern:node_name)
```

For example:

```
.probe ixba(x0.xx5.*:x0.a5)
.measure tran avg_xp avg ixba(x0.xx5.*:x0.a5)
```

The `ixba()` probe statement is equivalent to `probe_waveform_ixba -ipattern x0.xx5.* -node x0.a5`. You can use the `ixba(ipattern:node)` function to access the signal if you need to use it for analysis statements.

### Examples

The following example shows how to use the `probe_waveform_ixba` command to replace ISUB functionality. ISUB is often used to probe the port current into a subcircuit. For more information about the ISUB functionality, see the *CustomSim User Guide*.

Because the XBA flow flattens the hierarchy of the SPF file, ISUB cannot be without `probe_waveform_ixba`. For example:

```
probe_waveform_ixba -ipattern X1.X2.X3 -node VDD
```

In [Figure 2](#), the green box represents the original subcircuit structure. The XBA flow is enabled on instance `xtop.x1.xa` with `load_ba_file -file xafile.spf -xba 1`.

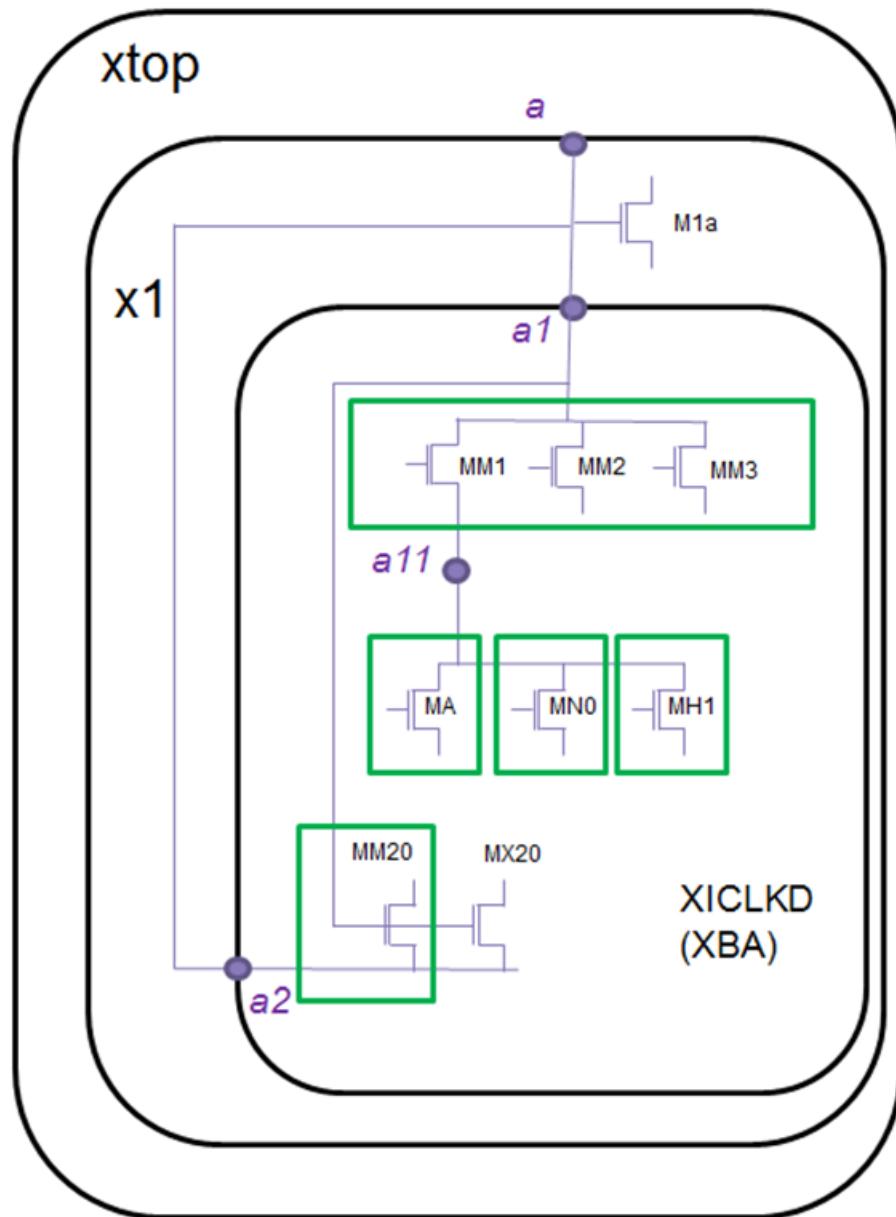


Figure 2 ISUB Example

The `xafile.spf` file has the following content:

## Chapter 2: CustomSim Batch Command Syntax

```

...
Instance Section
X1/CLKD/X1/XCKGD1/MM1...
XI/CLKD/X1/XCKGD1/MM2...
XI/CLKD/X1/XCKGD1/MM3...
XI/CLKD/X1N/XPLL/XCLK/MM20...
X1/CLKD/MX20...
...
X1/CLKD/X2/X1NAND/MA...
X1/CLKD/X1N202 [5]/MNO...
X1/CLKD/XXXTIEL/MH1...
...

```

The following command traces all the devices below the hierarchy of `xtop.x1.xiclkd` touching the node `a` and probes the sum of all the device terminal currents below the hierarchy `xtop.x1.xiclkd`.

```
probe_waveform_ixba -ipattern xtop.x1.xiclkd.* -node a
```

The CustomSim tool probes the sum of device current terminal of the following devices.

```
Ixba(xtop.x1.xiclkd :a) = Iterminal(XI/CLKD/X1/XCKGD1/MM1)
+ Iterminal(XI/CLKD/X1/XCKGD1/MM2)
+ Iterminal(XI/CLKD/X1/XCKGD1/MM3)
+ Iterminal(XI/CLKD/XIN/XPLL/XCLK1/MM20)
+ Iterminal(XI/CLKD/MX20)
```

Where:

$$\begin{aligned} I_{\text{terminal}}(\text{XI/CLKD/XIN/XPLL/XCLK1/MM20}) &= \\ &I_g(\text{XI/CLKD/XIN/XPLL/XCLK1/MM20}) \\ &+ I_s(\text{XI/CLKD/XIN/XPLL/XCLK1/MM20}) \end{aligned}$$

$$I_{\text{terminal}}(\text{XI/CLKD/MX20}) = I_g(\text{XI/CLKD/MX20}) + I_s(\text{XI/CLKD/MX20})$$

In reference to ISUB, the current output from `probe_waveform_xba` is slightly difference from ISUB. Originally if XBA is not enabled and ISUB is used:

```
Isub(xtop.a) = Iterminal(XI/CLKD/X1/XCKGD1/MM1)
+ Iterminal (XI/CLKD/X1/XCKGD1/MM2)
+ Iterminal (XI/CLKD/X1/XCKGD1/MM3)
+ Iterminal (XI/CLKD/XIN/XPLL/XCLK1/MM20)
+ Iterminal (XI/CLKD/MX20)
+ Iterminal (M1a)  
  

Isub(xtop.x1.a1) = Iterminal(XI/CLKD/X1/XCKGD1/MM1)
+ Iterminal (XI/CLKD/X1/XCKGD1/MM2)
+ Iterminal (XI/CLKD/X1/XCKGD1/MM3)
+ Ig (XI/CLKD/XIN/XPLL/XCLK1/MM20)
+ Ig (XI/CLKD/MX20)
```

The following command traces all the devices below the hierarchy of `xtop.x1.xiclkd.x2` touching the node `xtop.x1.xiclkd.a11`. Based on the SPF file, there are six devices touching node `xtop.x1.xa.a11`, but there is only one device below the hierarchy `xtop.x1.xiclkd.x2`.

```
probe_waveform_ixba -ipattern xtop.x1.xiclkd.x2.* -node
xtop.x1.xiclkd.a11
```

The outcome of this command is the current terminal of `XI/CLKD/X2/XINAND/MA`:

```
Ixba(xtop.x1.xiclkd.a11) = Iterminal(XI/CLKD/X2/XINAND/MA)
```

You can specify:

```
probe_waveform_ixba -ipattern xtop.x1.xiclkd.mn* -node a
```

To probe current from devices with name starting from `xtop.x1.xiclkd.mn`, or:

```
probe_waveform_ixba -ipattern xtop.x1.*.* -node a
```

To probe currents from all devices under hierarchy `xtop.x1.*` (third level of hierarchy) that are connected to node `a`.

### See Also

[load\\_ba\\_file](#), [probe\\_waveform\\_logic](#), [probe\\_waveform\\_va](#),  
[probe\\_waveform\\_voltage](#), [setWildcardRule](#)

## probe\_waveform\_logic

Generates logic values of specified nodes in the output waveform file.

### Syntax

```
probe_waveform_logic -node node {node}
    [-nN inst_name {inst_name}]
    [-nall inst_name {inst_name}]
    [-loth low_threshold]
    [-hith high_threshold]
    [-subckt subckt_name {subckt_name}]
    [-limit level]
    [-level level_val]
    [-except_node node_name {node_name}]
    [-port enable_value]
    [-filetag file_tag]
```

Argument	Description
-node node {node}	Specifies the nodes to be printed as a logical signal.
-nN inst_name {inst_name}	Specifies the name of the instance. The Nth terminal is probed as a logic signal, where N is a positive integer. Note that this option cannot probe subcircuit instance terminals.
-nall inst_name {inst_name}	Specifies the name of the instance in which all terminals are probed as a logic signal. Note that this option cannot probe subcircuit instance terminals.
-loth low_threshold	Specifies the threshold voltage for the LOW logic state, 0.
-hith high_threshold	Specifies the threshold voltage for the HIGH logic state, 1.
-subckt subckt_name {subckt_name}	Probes the nodes of the specified subcircuits as logic signals.

Argument	Description
<code>-limit limit_val</code>	Specifies the hierarchy level down to which node logic values are probed when you use wildcard characters. A value of 0 specifies the top level. The default for <code>limit_val</code> is 3. When you specify <code>-subckt</code> , <code>limit_val</code> is relative to the scope of <code>subckt_name</code> . See <a href="#">Table 5</a> for probing behavior with different combinations of <code>-limit</code> and <code>-level</code> .
<code>-level level_val</code>	Specifies the relative hierarchical level down to which node logic values are probed when you use wildcard characters. Numbering starts with 1 at the hierarchical level where the first wildcard appears. When you specify <code>-subckt</code> , <code>level_val</code> is relative to the scope of <code>subckt_name</code> in addition to the position of the first wildcard. See <a href="#">Table 5</a> for probing behavior with different combinations of <code>-limit</code> and <code>-level</code> .
<code>-except_node node_name {node_name}</code>	Specifies the nodes to exclude from probing.
<code>-port enable_value</code>	If you specify an <code>enable_value</code> value of 1, a wildcard character matches the subcircuit port names.
<code>-filetag file_tag</code>	Specify a file tag to direct the signals probed with this command to a separate waveform file. The waveform file has the standard name with <code>.filetag</code> inserted before the normal file suffix, for example, <code>xa.filetag.wdf</code> .

### Description

The default settings for `-loth` and `-hith` are derived from [set\\_logic\\_threshold](#). By default, `-loth` is set to 30% of the low voltage, and `-hith` is set to 70% of the high voltage. The CustomSim tool uses a search algorithm to determine the high/low voltage for a given node if there are multiple supply domains:

- If only `-loth` is specified, a voltage  $\leq$  the `low_threshold` signal is 0, else 1.
- If only `-hith` is specified, a voltage  $\geq$  the `high_threshold` signal is 1, else 0.
- If `-loth` and `-hith` are both specified:

- If  $-loth \geq -hith$ , error out.
- If voltage  $< low\_threshold$ , the signal is 0.
- If voltage  $\geq high\_threshold$ , the signal is 1.
- If  $low\_threshold < voltage < high\_threshold$ , the signal is U (unknown).

Different signals can use different logic thresholds; but for a single signal, there can be only one logic threshold, even if the signal is being probed to several file tags as specified by the **-filetag** option.

**Note:** You can only apply the **-nN** and **-nall** arguments to primitive instances.

### Examples

```
probe_waveform_logic -node data -loth 0.6
```

This example prints the logic state of the data node in the waveform file. If data has a voltage of  $\leq 0.6V$ , the logic state is 0; otherwise, the logic state is 1.

### See Also

[probe\\_waveform\\_current](#), [probe\\_waveform\\_va](#), [probe\\_waveform\\_voltage](#), [set\\_wildcard\\_rule](#)

---

## probe\_waveform\_va

Probes the values of Verilog-A variables or parameters and writes them to the plot file.

### Syntax

```
probe_waveform_va -var variable_name {variable_name}
    [-subckt subckt_name]
    [-limit level]
    [-level level_val]
```

---

Argument	Description
<code>variable_name</code> <code>{variable_name}</code>	Specifies the hierarchical path to the Verilog-A variable.

---

Argument	Description
<code>-subckt subckt_name</code>	Specifies to probe the variables in all instances of the specified subcircuit. If you specify <code>-subckt</code> the <code>var_name</code> is local to the subcircuit.
<code>-limit limit_val</code>	Specifies the hierarchy level down to which values of Verilog-A variables and parameters are probed when you use wildcard characters. A value of 0 specifies the top level. The default for <code>limit_val</code> is 3. When you specify <code>-subckt</code> , <code>limit_val</code> is relative to the scope of <code>subckt_name</code> . See <a href="#">Table 5</a> for probing behavior with different combinations of <code>-limit</code> and <code>-level</code> .
<code>-level level_val</code>	Specifies the relative hierarchical level down to which values of Verilog-A variables and parameters are probed when you use wildcard characters. Numbering starts with 1 at the hierarchical level where the first wildcard appears. When you specify <code>-subckt</code> , <code>level_val</code> is relative to the scope of <code>subckt_name</code> in addition to the position of the first wildcard. See <a href="#">Table 5</a> for probing behavior with different combinations of <code>-limit</code> and <code>-level</code> .

### Description

Note that `probe_waveform_va`:

- Probes the value of Verilog-A variables or parameters and writes them to the plot file. It does not probe Verilog-A ports or electrical nodes. Use the [`probe\_waveform\_voltage`](#) command to probe electrical signals.
- Scopes only to Verilog-A instances/modules. It does not print SPICE MOS model parameters.

The output to the waveform file is the hierarchical path and the variable name. There are no signal access functions such as `v()` or `i()` for the voltage and current signals. The separator between the hierarchical path and the variable name is the hierarchical separator for the simulation as defined by the [`set\_sim\_hierid`](#) command or the `.hier` command (Eldo format only).

The CustomSim tool-supported wildcard characters ( \* ) can be used in the `variable_name` identifier. The wildcard character can be used to match one level, or all levels, of the hierarchy. See the [`setWildcardRule`](#) command.

## Examples

```
probe_waveform_va -var x1.count
```

Probes the variable count in the x1 module. The variable appears in the waveform file as x1.count.

```
probe_waveform_va -var x1.count
```

Probes the variable count in the x1 module. The variable appears in the waveform file as x1:count.

```
probe_waveform_va -var x1.*
```

Probes all variables and parameters in the x1 module.

```
probe_waveform_va * -subckt mymodule
```

Probes all variables and parameters in all instances of mymodule.

## See Also

[probe\\_waveform\\_current](#), [probe\\_waveform\\_logic](#),  
[probe\\_waveform\\_voltage](#), [set\\_wildcard\\_rule](#)

---

## probe\_waveform\_voltage

Creates a voltage waveform output.

### Syntax

```
probe_waveform_voltage -v node_name {node_name}
[-vn instance_name {instance_name}]
[-vall instance_name {instance_name}]
[-vsub subckt_instance_name.port
{subckt_instance_name.port}]
[-subckt subckt_name]
[-limit limit_val]
[-level level_val]
[-except_node node_name {node_name}]
[-port enable_value]
[-ba_net net_name {net_name}]
[-filetag file_tag]
```

Argument	Description
<code>-v node_name{ node_name}</code>	Specifies the node name at which the voltage is probed.
<code>-vn instance_name</code>	Specifies the name of the instance at which the voltage of terminal <i>n</i> is probed, where <i>n</i> is a positive integer.
<code>-vall instance_name</code>	Specifies the name of the instance at which the voltage of all terminals are probed.
<code>-vsub subckt_instance_name.port {subckt_instance_name.port}</code>	Probes the ports of the subcircuit using local port names. You can use wildcard characters. This probe matches ports identically to an isub probe, except that it reports voltage instead of current.
<code>-subckt subckt_name</code>	Probes the nodes in all instances of the named subcircuit. If you specify this argument, the nodes or instances should be in the subcircuit instance hierarchy level.
<code>-limit limit_val</code>	Specifies the hierarchy level down to which voltages are probed when you use wildcard characters. A value of 0 specifies the top level. The default for <i>limit_val</i> is 3. When you specify <code>-subckt</code> , <i>limit_val</i> is relative to the scope of <i>subckt_name</i> . See <a href="#">Table 5</a> for probing behavior with different combinations of <code>-limit</code> and <code>-level</code> .
<code>-level level_val</code>	Specifies the relative hierarchical level down to which voltages are probed when you use wildcard characters. Numbering starts with 1 at the hierarchical level where the first wildcard appears. When you specify <code>-subckt</code> , <i>level_val</i> is relative to the scope of <i>subckt_name</i> in addition to the position of the first wildcard. See <a href="#">Table 5</a> for probing behavior with different combinations of <code>-limit</code> and <code>-level</code> .

Argument	Description
<code>-except_node node_name {node_name}</code>	Specifies the nodes to exclude from probing.
<code>-port enable_value</code>	If you specify an enable_value value of 1, a wildcard character matches the subcircuit port names.
<code>-ba_net net_name {net_name}</code>	Probes the voltage waveforms for the SPF/SPEF pin names. You can specify a wildcard character in a net name. Note that this argument only works in back-annotation and is not applicable to the <code>-subckt</code> and <code>-limit</code> arguments.
<code>-filetag file_tag</code>	Specify a file tag to direct the signals probed with this command to a separate waveform file. The waveform file has the standard name with <code>.filetag</code> inserted before the normal file suffix, for example, <code>xa.mytag.wdf</code> .

**Description**

Probes the voltage on a node or on the pin of a primitive instance. The voltage waveform is written to the output file in the format specified by the `post` option in the netlist.

The CustomSim tool-supported wildcard character ( `*` ) can be used in the `node_name` and `instance_name` identifiers. The wildcard character can be used to match one level—or all levels—of the hierarchy. See the [set\\_wildcard\\_rule](#) command.

Probes specified by this command are in addition to the `.probe` statement in the HSPICE or Eldo netlist files or `save` statements in the Spectre netlist files.

Long simulations, or simulations in which some nodes have a high level of activity, can produce very large waveform files. To minimize waveform file loading time in these files, you can direct signals to separate waveform files and keep file sizes smaller.

To direct signals to a separate waveform file, use the `-filetag` argument. All of the signals probed by that instance of the command are directed to a separate waveform file. The file name has the same format as the standard waveform file name, except for the suffix; for example, `xa.mytag.wdf`. All of the settings made by [set\\_waveform\\_option](#) also apply to the tagged output

files. A signal can be directed to multiple files if it is probed with another command that specifies a different *file\_tag*. Additionally, multiple `probe_waveform_voltage` commands can use the same file tag. The other probing commands (voltage and logic) can also use the same file tag.

### ***Understanding the Difference Between -level and -limit***

The `-level` and `-limit` option are relative to the current hierarchy level set by `-subckt`. When you use a `*` is wildcard character, `-level` is relative to the hierarchical level at which first `*` appears.

*Table 5 -limit and -level Combinations*

	<b>-limit</b>	<b>-level</b>	<b>Comment</b>
1	Not specified	Not specified	Default with -limit 3
2	Specified	Not specified	-limit takes effect
3	Not specified	Specified	-level takes effect
4	Specified	Specifie	The more restrictive one between -level and -limit takes effect

For example:

```
probe_waveform_voltage * -limit 0
```

Probes all nets at the top level only. It is the same as:

```
probe_waveform_voltage * -level 1
```

The following example probes all nets at the top level and 1 level of hierarchy below.

```
probe_waveform_voltage * -limit 1
```

It is the same as:

```
probe_waveform_voltage * -level 2
```

The following example shows no probes because limit 0 is the top level.

```
probe_waveform_voltage x1.* -limit 0
```

The following example shows no probes because limit 1 is the first level of hierarchy from the top, and `x1.x2.*` is the second level of hierarchy, so a minimum limit of 2 is required.

```
probe_waveform_voltage x1.x2.* -limit 1
```

The following example probes all nets in `x1` (but does not probe in `x1.x2`, `x1.x3`, and so on).

```
probe_waveform_voltage x1.* -level 1
```

It is the same as:

```
probe_waveform_voltage x1.* -limit 1
```

The following example probes all nets in `x1.x2` (but does not probe in `x1.x2.x1`, `x1.x2.x3`, and so on).

```
probe_waveform_voltage x1.x2.* -level 1
```

It is the same as:

```
probe_waveform_voltage x1.x2.* -limit 2
```

## Examples

### *Example 8*

```
probe_waveform_voltage *
```

[Example 8](#) probes all voltage nodes down to the default level of hierarchy, which is 3.

### *Example 9*

```
probe_waveform_voltage *.* -limit 4
```

[Example 9](#) probes all voltage nodes from hierarchy level 1 to level 4.

### *Example 10*

```
probe_waveform_voltage *.* -limit 0
```

[Example 10](#) does not probe the voltage of any node. The asterisk characters `*.*` refer to all of the nodes at the first level and below, but the `-limit 0` argument limits the nodes (to be probed) to only the top level. The arguments are, therefore, contradictory.

*Example 11*

```
probe_waveform_voltage * -except_node *clk
```

**Example 11** probes all nodes down to the second level of hierarchy, except those finishing with `clk` pattern.

*Example 12*

```
probe_waveform_voltage -ba_net clk*
```

**Example 12** probes the voltage waveform from the SPF/SPEF instance pins for any net name that matches `clk`.

*Example 13*

```
probe_waveform_voltage ctrl sig* -subckt mysub -limit 2
```

**Example 13** specifies to probe the `ctrl` signal and all signals that begin with `sig` in all occurrences of the `mysub` subcircuit. The `-limit 2` option is relative to the top level of the subcircuit and limits the depth of the wildcard probe. **Example 13** is equivalent to the following SPICE definition:

```
.subckt mysub
.probe v(ctrl)
.probe v(sig*) level=2
...
.ends
```

*Example 14*

```
probe_waveform_voltage rx_data -subckt rxblock -filetag rx_block
```

**Example 14** probes the signal `rx_data` in all instances of `rxblock` and diverts them to the `rx_block` waveform file. If the CustomSim tool is invoked as `xa net.sp -o xa -wavefmt wdf`, the waveform file is named `xa.rx_block.wdf`.

The following examples use this sample circuit:

*Example 15*

```
.subckt topsub t1 t2
X1 t1 t2 downsub
...
.ends
.subckt downsub d1 d2
...
.ends
X0 a b topsub
X01 c d topsub
```

```
probe_waveform_voltage -vsub x0.* -limit 1
```

Probes all of the ports in `xtopsub` and any subcircuits contained in `topsub` down to the limit of 1. The following probes are in the waveform file: `v(x0.t1)`, `v(x0.t2)`.

```
probe_waveform_voltgae -vsub x0.*
```

Probes all of the ports in `x0` and any subcircuits contained in `x0` down to the limit of 1. The following probes are in the waveform file: `v(x0.t1)`, `v(x0.t2)`, `v(x0.x1.d1)`, and `v(x0.x1.d2)`.

You can combine the `vsub` probe with the `-subckt` option. The `vsub` instance.port becomes local to the subcircuit definition:

```
probe_waveform_voltage -vsub * -subckt topsub
```

This command probes all subcircuit ports in all instances of the named subcircuit as well as the ports of any subcircuit instance in `topsub` down the default limit of 3. This example probes: `v(x0.t1)`, `v(x0.t2)`, `v(x0.x1.d1)`, `v(x0.x1.d2)`, `v(x01.t1)`, `v(x01.t2)`, `v(x01.x1.d1)`, and `v(x01.x1.d2)`.

To probe only the ports of the named subcircuit use `-limit 0` to prevent matching any deeper into the hierarchy:

```
probe_waveform_voltage -vsub * -subckt topsub -limit 0
```

This command probes all subcircuit ports in all instances of the named subcircuit, `topsub`: `v(x0.t1)`, `v(x0.t2)`, and `v(x01.t1)`, `v(x01.t2)`.

```
probe_waveform_voltage -vsub * -subckt downsub -limit 0
```

Probes only the port voltages of all instances of the `downsub` subcircuit. The following probes are in the waveform file: `v(x0.x1.d1)`, `v(x0.x1.d2)`, `v(x01.x1.d1)`, and `v(x01.x1.d2)`.

When you use wildcard matching except patterns can be used to exclude some nodes:

```
probe_waveform_voltage -vsub * -subckt downsub -limit 0 -  
except_node x0.*
```

This command probe only the ports of all instances of the `downsub` subcircuit, but excludes any instance of `downsub` in `x0`. The following probes are added to the waveform file: `v(x01.x1.d1)`, `v(x01.x1.d2)`.

#### *Example 16*

```
probe_waveform_voltage x1.x2.* -level 4
```

Probes all voltage nodes from hierarchy level 2 to level 5.

*Example 17*

```
probe_waveform_voltage * -limit 2 -level 6 -subckt volgen
```

Probes all voltage nodes from hierarchy level 0 to level 2 inside subcircuit volgen.

**See Also**

[probe\\_waveform\\_current](#), [probe\\_waveform\\_logic](#), [probe\\_waveform\\_va](#),  
[set\\_wildcard\\_rule](#)

## pulse\_oscillator

Applies a current kick to a specified node.

**Syntax**

```
pulse_oscillator -node node_name {node_name}
    -pw pulse_width
    -time value [value ...]
    [-amp amp_value]
    [-rt rt_value]
```

Argument	Description
-node <i>node_name</i> { <i>node_name</i> }	Defines the node at which the current source is connected.
-pw <i>pulse_width</i>	Specifies the duration of the current pulse.
-time <i>value</i>	Specifies the time at which the current pulse starts. You can specify multiple pulses by listing multiple times.
-amp <i>amp_value</i>	Specifies the amplitude of the current pulse.
-rt <i>rt_value</i>	Specifies the rise and fall time of the current pulse.

**Description**

A current pulse with a pulse width of half the expected oscillation period is usually sufficient to start oscillations, but sometimes you need to experiment with the pulse amplitude.

If a circuit has a fully differential structure, then use two `pulse_oscillator` commands: one connected to the positive and negative branch of the differential structure and one applied with opposite polarity.

### Examples

```
pulse_oscillator -node xosc.pl -pw 1n -time 10u -amp 1u  
pulse_oscillator -node xosc.nl -pw 1n -time 10u -amp -1u
```

---

## release\_node\_voltage

Releases the node voltages from the values fixed by [force\\_node\\_voltage](#). When you specify this command, the simulation results determine the node voltages.

### Syntax

```
release_node_voltage -node node_name {node_name}  
[-time time_value]  
[-subckt subckt_name]
```

---

Argument	Description
-node <i>node_name</i> { <i>node_name</i> }	Specifies the node names to release.
-time <i>time_value</i>	Specifies the time to release the nodes. The default is 0.
-subckt <i>subckt_name</i>	Releases only the nodes inside the specified subcircuit.

---

### Examples

```
release_node_voltage vpump -time 150ns
```

The `vpump` signal previously forced to a given value is released at 150ns. The simulation results determine the `vpump` voltage value until the end of the run unless you use a new [force\\_node\\_voltage](#) command.

## report\_dangling\_node

Reports dangling nodes in a separate file.

### Syntax

```
report_dangling_node enable_value
```

Argument	Description
enable_value	By default, when this option is off, the CustomSim tool reports dangling nodes in the log file. Turn this option on to report dangling nodes in a separate file with the following format: <i>sim_output_file_name.dng</i> .

## report\_floating\_node

Reports all floating nodes in a file with a .fnode extension, or .fgate if you only request floating gates.

### Syntax

```
report_floating_node [-type type]
                     [-format format]
                     [-limit limit]
                     [-file file_name]
```

Argument	Description
-type type	Can specify: <ul style="list-style-type: none"> <li>▪ gate generates a list of floating gates in a .fgate file.</li> <li>▪ all generates a list of floating nodes in a .fnode file (default).</li> </ul>
-format format	Can specify: <ul style="list-style-type: none"> <li>▪ ic specifies the format in the form of a .ic statement (default).</li> <li>▪ vsrc specifies the format in the form of voltage source.</li> <li>▪ report specifies the format in the form of node and port connection table.</li> </ul>

Argument	Description
<code>-limit limit</code>	Specifies the hierarchy level down to which a floating node is checked and reported. You can only use this option with <code>-format report</code> . By default, all floating nodes are reported.
<code>-file file_name</code>	Specifies the name of the output file that contains the floating node information. It has a <code>.fnode</code> extension if you use <code>-type all</code> and a <code>.fgate</code> extension if you use <code>-type gate</code> .

### Description

A floating node is a node that has no DC path to ground and touches at least one of the following:

- The gate of a MOSFET.
- A current source.
- The controlling input node of controlled source

Floating gates are the most common floating nodes.

**Note:** You cannot use this command with the `set_floating_node` command. If you specify both the `set_floating_node` and `report_floating_node` commands, then the last one is used.

### Examples

```
report_floating_node -type all -format report
```

Specifies a report that includes all floating nodes, including floating bulks. The report is in the form of a table.

```
report_floating_node -type all -format ic
```

Specifies a report that includes all floating nodes, except floating bulks. The report is in the form of a `.ic` statement.

## report\_model

Reports detailed model information. This command lets you generate reports that contain detailed model information similar to what the Eldo format provides in the .chi file.

### Syntax

```
report_model -report report_value
    [-generate enable_value]
    [-stop enable_value]
    [-count model|subckt|all|none]
```

Argument	Description
<code>-report <i>report_value</i></code>	Specifies one of the following values: <ul style="list-style-type: none"><li>▪ 1 to generate a condensed report. DEVICE, MODEL, TYPE, LEVEL, and INSTANCE information is provided, but the detailed list of model parameters is suppressed.</li><li>▪ 2 to generate a full report, including detailed model parameters.</li><li>▪ 3 to generate a full report with all devices (not only devices that have a .model) and all instances, even when they use identical models.</li></ul>
<code>-generate <i>enable_value</i></code>	Generates a netlist that contains .model cards and a single instantiation of each unique model instance. None of the original circuit netlist connectivity is in the file. This information is written to a file with a .model extension.
<code>-stop <i>enable_value</i></code>	Stops the simulation after front-end processing before starting DC or transient analysis. This argument can provide a quick analysis to help debug models.

---

<b>Argument</b>	<b>Description</b>
-count model subckt all  none	<p>Specify one of the following options to control how to report the list of models.</p> <ul style="list-style-type: none"> <li>▪ model to report the list of all models and their count in the <code>.modelinfo</code> file.</li> <li>▪ subckt to report the list of all subcircuits and their count in the <code>.modelinfo</code> file.</li> <li>▪ all to report the list of all models and subcircuits and their count in the <code>.modelinfo</code> file.</li> <li>▪ none (default) to not report model and subcircuit information in the <code>.modelinfo</code> file.</li> </ul>

---

## **report\_node\_alias**

Reports all the aliased nodes in the report files.

### **Syntax**

```
report_node_alias -hierarchy enable_value | -short
enable_value
```

---

<b>Argument</b>	<b>Description</b>
-hierarchy <i>enable_value</i>	If set to 1, reports all hierarchy node aliases to a file. The default is 0, which does not report hierarchy node aliases.
-short <i>enable_value</i>	If set to 1, reports all hierarchy node aliases to a file. The default is 0, which does not report shorted node aliases.

---

### **Description**

This command prints the nodes that have aliased node names. A node might have an alias because of:

- Hierarchical alias names from instances of subcircuits.
- Shorted alias names due to very small resistors or a DC voltage source of 0 volt connected to a node.

The primary node name is the node name that remains in the database. The alias node name is the name that is removed from database. The alias output file has two or more columns such as:

```
<primary_node_name> <alias_node_name_0> ...
<alias_node_name_n>
```

### Examples

```
report_node_alias -hierarchy 1
```

Reports all hierarchy alias nodes in *output\_file.nodealias*.

```
report_node_alias -short 1
```

Reports all shorted nodes in *output\_file.nodealias*.

```
report_node_alias -hierarchy 1 -short 1
```

Reports all hierarchy alias nodes and all shorted nodes in *output\_file.nodealias*.

## report\_node\_cap

Reports capacitance information for the specified nodes.

### Syntax

```
report_node_cap -node node_name {node_name}
    [-short_resistor value]
    [-group group_name]
    [-limit limit_value]
    [-report basic|detail]
```

Argument	Description
-node <i>node_name</i> { <i>node_name</i> }	Reports capacitance for the node names you specify. You can use wildcard characters in the node names.
-short_resistor <i>value</i>	Specifies the value of the resistor to be shorted. The default value is 0, which means none of the resistors are shorted.
-group <i>group_name</i>	Creates a group name for the nodes you specify with -node. If you specify this option, all nodes for a report_node_cap command are grouped together. the CustomSim tool reports the capacitance information based on this group. Use this option only for a flat, postlayout design.

Argument	Description
-limit limit_value	Specifies the hierarchy level down to which the CustomSim tool reports the capacitance information. The default is 3.
-report basic detail	Specifies the type of report to print. Use the <code>basic</code> keyword (the default) to print only the basic capacitance information. Use the <code>detail</code> keyword to print a detailed report.

**Description**

The reported node capacitance information includes total node capacitance, wire capacitance, gate capacitance of a MOSFET, and junction capacitance of a MOSFET. You can specify multiple commands in a simulation. the CustomSim tool processes each command separately.

`report_node_cap` outputs the capacitance information in a `*.cap#` file.

In a prelayout design, capacitance is reported as:

$$C_{total} = C_{gate} + C_{junction} + C_{wire}$$

$$C_{gate} = C_{gd} + C_{gs} + C_{gb}$$

$$C_{junction} = C_{db} + C_{sb}$$

$$C_{wire} = C_{design}$$

In postlayout design, the CustomSim tool expands a single node from the prelayout design into multiple nodes because of the RC parasitics. The postlayout flow is divided into 2 scenarios:

- A back-annotated postlayout.
- A flat postlayout.

In the back-annotated postlayout flow, you can trace the connectivity back to the prelayout design with the information from the prelayout netlist and the back-annotation file:

```
* | NET na 0.00458507PF <-- Net Capacitance (CBAnet)
* | I (x02/mp:GATE x02/mp GATE I 4.8e-16 22.75 3.25) //
$llx=22.55 $lly=3.25 $urx=22.95 $ury=3.25 $lvl=5
* | I (x02/mn:GATE x02/mn GATE I 2.4e-16 22.75 1.05) //
$llx=22.55 $lly=0.6 $urx=22.95 $ury=1.05 $lvl=4
* | I (x01/mp:DRN x01/mp DRN B 0 8.45 3.25) // $llx=8.45
$lly=2.65 $urx=9.2 $ury=3.85 $lvl=7
```

```

*|I (x01/mn:DRN x01/mn DRN B 0 8.45 1.05) // $llx=8.45
$lly=0.75 $urx=9.2 $ury=1.35 $lvl=6

*|S (na:1 22.75 2.65) // $llx=22.55 $lly=2.65 $urx=22.95
$ury=2.65 $lvl=3

*|S (na:2 22.75 3.925) // $llx=22.55 $lly=3.85 $urx=22.95
$ury=4 $lvl=5

Cg1 na:1 0 4.33011e-17

Cg2 na:2 0 3.99892e-17

...
R158 na:1 x02/mp:GATE 4.8 $l=0.6 $w=0.4 $lvl=5
R159 na:1 na:3 30 $l=0.8 $w=0.4 $lvl=3
R160 x02/mp:GATE na:2 5.016 $l=0.675 $w=0.4 $lvl=5
R161 na:3 na:4 16.875 $l=0.5 $w=0.4 $lvl=3
R162 na:3 na:5 18.75 $l=0.5 $w=0.4 $lvl=3
R163 na:3 na:8 3.96 $a=0.04 $lvl=10
R164 na:4 na:6 3.96 $a=0.04 $lvl=1

...
R186 na:18 x01/mn:DRN 5.38888 $a=0.09 $lvl=12
R187 na:19 na:20 0.992002 $l=0.8 $w=0.5 $lvl=2
R188 na:19 na:21 13.1234 $l=6.45 $w=0.3 $lvl=2
R189 na:20 na:24 0.744002 $l=0.6 $w=0.5 $lvl=2
R190 na:20 x01/mp:DRN 5.38888 $a=0.09 $lvl=12

...

```

In back-annotated post-layout flow, the capacitance information is reported as:

$C_{total} = C_{gate} + C_{junction} + C_{wire}$

$C_{gate} = C_{gd} + C_{gs} + C_{gb}$

$C_{junction} = C_{db} + C_{sb}$

$C_{wire} = C_{BAnet} + C_{design}$

Based on the previous back-annotation file example:

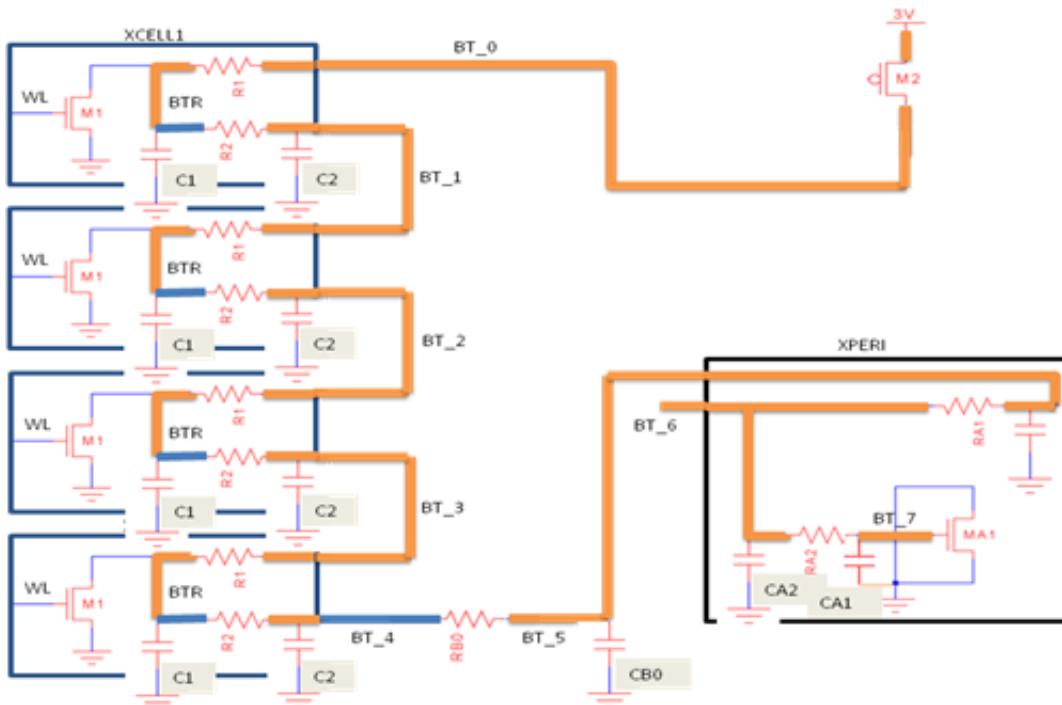
$C_{total} = C_{gate} + C_{junction} + C_{wire}$

$$\text{Cgate} = \text{Cgate}(x02/mp) + \text{Cgate}(x02/mn)$$

$$\text{Cjunction} = \text{Cjunction}(x01/mp) + \text{Cjunction}(x01/mn)$$

$$\text{Cwire} = \text{CBAnet}(0.00458507\text{PF}) + \text{Cdesign}$$

In a flat postlayout flow, the CustomSim tool treats all nodes as unique and independent. To accurately calculate the capacitance information, you need to group the nodes and then run `report_node_cap`. For example, see [Figure 3](#).



*Figure 3 Flat Postlayout Flow Example*

In [Figure 3](#), a prelayout node, `BT`, has been expanded into different nodes in the postlayout. Because the postlayout netlist is flat, and there is no trace of connectivity from the prelayout netlist and back-annotation flow, all nodes are treated as unique and independent.

To accurately report the capacitance information, you need to tell the CustomSim tool which nodes can be grouped together for `report_node_cap` command to calculate the capacitance information. To group a list of nodes into one group, use `-group` argument and list the names of nodes to be grouped:

BT\_0, BT\_1, BT\_2, BT\_3, BT\_4, BT\_5, BT\_6, XPERI.BT\_7, XCELL1.BTR, XCELL2.BTR, XCELL3.BTR, and XCELL4.BTR and do the following steps:

1. Specify the following report\_node\_cap command.

```
report_node_cap -node BT_? XPERI.BT_7 XCELL?.BTR -group BT
```

This command groups all the specified node names into one group named BT.

2. Assuming all parasitic capacitor has a value of 1 fF, the CustomSim tool calculates the capacitance information is as:

$$C_{total} = C_{gate} + C_{junction} + C_{wire}$$

$$C_{gate} = C_{gate}(XPERI.MA1)$$

$$C_{junction} = C_{junction}(M2) + C_{junction}(XCELL1.M1) + \\ C_{junction}(XCELL2.M1)$$

$$+ C_{junction}(XCELL3.M1) + C_{junction}(XCELL4.M1) + \\ C_{junction}(XPERI.MA1)$$

$$C_{wire} = (12 * 1 \text{ fF}) + C_{design}$$

$$C_{design} = 0 \text{ pF}$$

## **report\_operating\_point**

Determines how the CustomSim tool exports initial conditions it computes in DC analysis. It also defines how the CustomSim tool reports the results of latch (and other circuit types) detection. The initial conditions are dumped in a file named *prefix.time.ic*. Note that you cannot specify multiple report\_operating\_point commands.

### **Syntax**

```
report_operating_point -time dc|end|time_value
  {dc|end|time_value}
  [-report all|core]
  [-file file_name]
  [-node_type node_type]
  [-type ic_type]
  [-subckt subckt {subckt}]
  [-inst inst {inst}]
  [-node_details switch_value]
```

Argument	Description
<code>-time dc end time_value</code>	Specifies the time to dump the operating point: <ul style="list-style-type: none"> <li>▪ <code>dc</code> for a time value of 0 (default).</li> <li>▪ <code>end</code> for the transient end time.</li> <li>▪ <code>time_value</code> for a user-defined time.</li> </ul>
<code>-report all core</code>	Specify <code>core</code> (the default) to dump the subset of the core nodes the CustomSim tool needs to determine the operating point. Specify <code>all</code> to force all nodes to be written to a file.
<code>-file file_name</code>	Adds operating point information to the specified file. The default file name is <code>prefix.time.ic</code> .
<code>-node_type node_type</code>	Limits the application of initial conditions to nodes detected as <code>node_type</code> . You can specify a list of arguments with either or both of the following keywords to limit the reporting: <ul style="list-style-type: none"> <li>▪ <code>latch</code> to report only the detected latches.</li> <li>▪ <code>memcell</code> to report only detected memory cells.</li> </ul>
<code>-type ic_type</code>	Specifies the type of initial condition to be applied. The <code>ic_type</code> can be <code>ic</code> or <code>nodeset</code> . This option overrides what is specified in the file.
<code>-subckt subckt {subckt}</code>	Limits the application of initial conditions to nodes of the instances of <code>subckt</code> . A list of arguments is supported. Wildcards are not supported.
<code>-inst inst {inst}</code>	Limits the application of initial conditions to the specified instances. Wildcards are supported.
<code>-node_details switch_value</code>	You can specify one of the following for <code>switch_value</code> : <ul style="list-style-type: none"> <li>▪ <code>0</code> (default) does not list node details.</li> <li>▪ <code>1</code> lists node details in a separate file. The file name is created by appending <code>.op_table</code> to the name of the initial conditions file (for example, <code>xa.0.ic.op_table</code>).</li> </ul>

## Examples

```
report_operating_point 0 -node_type latch -type nodeset
```

Writes out a *prefix.0.ic* file containing initial conditions from time 0 as nodeset for latch nodes only.

```
report_operating_point 10n -file 10n-op -subckt latch_fast -  
node_type latch
```

Writes out a *10n-op* file containing initial conditions from time 10n as ic only for latch nodes and nodes in instances of the *latch\_fast* subcircuit.

```
report_operating_point -time 0 -node_details 1
```

Writes out a *prefix.0.ic* file containing initial conditions from time 0 as ic for all nodes, including node details. The node details are in the *prefix.0.ic.op\_table* file.

```
report_operating_point 0 -node_type latch -type nodeset
```

Writes a file *<prefix>.0.ic* that contains initial conditions from time 0 as nodeset for latch nodes only.

```
report_operating_point 0 -node_type memcell -type nodeset
```

Writes a file *<prefix>.0.ic* that contains initial conditions from time 0 as nodeset for memory cell nodes only.

```
report_operating_point -time 0 -node_details 1
```

Produces a *file .0.ic* file that contains an operating point at time 0 with *.ic* for the core nodes and also generates a *.0.ic.op\_table* file with node initialization details.

---

## report\_power

Generates power consumption reports.

### Syntax

Reporting by port name:

```
report_power -port port_name {port_name}  
[-label label_name]  
[-twindow tstart [{tstop}] {tstart [tstop]})
```

```
[-limit level]
[-subckt subckt]
[-avg enable_value]
[-rms enable_value]
[-max enable_value]
[-min enable_value] [-probe enable_value]
[-except_port except_pattern]
```

Reporting by node name:

```
report_power -by_node node_name {node_name}
  [-label label_name]
  [-twindow tstart [{tstop}] {tstart [tstop]}]
  [-limit level] [-subckt subckt] [-avg enable_value]
  [-rms enable_value]
  [-max enable_value]
  [-min enable_value]
  [-probe enable_value]
  [-except_port except_pattern]
  [-report basic|detail]
```

---

Argument	Description
-port port_name {port_name}	Specifies the currents to report based on port names. A preceding “*” or “.” character in the port name is optional.
-by_node node_name {node_name}	Specifies the currents to report based on connectivity. You can use wildcards in the node names.
-label label_name	Specifies the text label for the report file.
-twindow tstart tstop {tstart tstop}	Performs the check within the time window defined by <i>tstart tstop</i> { <i>tstart tstop</i> }. The <i>tstart</i> and <i>tstop</i> must come in pairs, except for the final window where if <i>tstop</i> is not specified it is assumed to be the end of the simulation. The final <i>tstop</i> can also be the end or END keyword.
-limit level	Specifies the maximum hierarchical depth for the report. The default is 3.
-subckt subckt	Specifies a subcircuit name for current reporting. Note that you cannot use this argument with the -by_node argument.

Argument	Description
<code>-avg enable_value</code>	Specifies whether to report the average value. The default is 1 for printing the average value. See the <a href="#">Common Syntax Definitions</a> section for details about the <code>enable_value</code> argument.
<code>-rms enable_value</code>	Specifies whether to report the rms value. The default is 1 for printing the rms value. See the <a href="#">Common Syntax Definitions</a> section for details about the <code>enable_value</code> argument.
<code>-max enable_value</code>	Specifies whether to report the maximum value. The default is 1 for printing the maximum value. See the <a href="#">Common Syntax Definitions</a> section for details about the <code>enable_value</code> argument.
<code>-min enable_value</code>	Specifies whether to report the minimum value. The default is 1 for printing the minimum value. See the <a href="#">Common Syntax Definitions</a> section for details about the <code>enable_value</code> argument.
<code>-probe enable_value</code>	Enables or disables the probing of ports: 1, true, or on to enable; 0, false, or off to disable. The default is 0.
<code>-except_port</code> <code>except_pattern</code>	Excludes ports that match the <code>except_pattern</code> from the report. You can specify wildcards in the pattern name.
<code>-report basic detail</code>	If <code>detail</code> is set, the following information is printed: Sub-circuit Level: xx Percentage of top sub-circuit: Avg (%) = xx Rms (%) = xx Percentage of parent sub-circuit: Avg (%) = xx Rms (%) = xx The default is <code>basic</code> . This argument is only supported with <code>-by_node</code> .

### Description

`report_power` produces a text report file that contains subcircuit port currents. There are two methods for specifying the reporting ports: by the port name or by connectivity. The report appears in the `file.power` file. The

`-limit` argument specifies the absolute hierarchical depth. The top level of the netlist hierarchy is 0.

To specify reporting by port name, use the `-port` argument. A subcircuit port name is defined by a subcircuit definition (`.subckt subname portname1 portname2 ...`). A leading “`*`” wildcard and “`.`” hierarchical delimiter are optional. You can use the `-subckt` argument to scope specific subcircuits. You can also use the `-except_port` argument to exclude some ports. Any port names matching the exclude pattern are not reported.

To specify reporting by connectivity, use the `-by_node` argument. The ports/terminals of the instances connected to the named nodes have their power reported. The CustomSim tool automatically creates a port/terminal list based on the connecting subcircuit and ideal voltage and current sources to the specified nodes. The list of ports and terminals traverses the hierarchy based on the `-limit` option. A power/current report is not generated for any primitive element other than ideal voltage and current sources. MOS, BJT, resistors, capacitors, and so on are excluded from the power/current report. Subcircuit instances that the CustomSim tool detects as a MOS macro model are also excluded. You can use the `-except_port` argument to exclude some ports. Any port names matching the exclude pattern are not reported.

A warning is issued when any of the excluded elements do connect to the specified node. The warning message is consistent with other CustomSim tool warnings. By default, only the first 10 warning are printed and can be controlled with the [set\\_message\\_option](#) command.

If `-by_node` is applied, the report is generated for those subcircuits/elements at that level of hierarchy, even if the hierachal level is exceeded in the `-limit` value. Additional levels of hierarchy are only included if they meet the `-limit` value. If a wildcard is used in the `-by_node` pattern, the `-limit` setting limits the hierarchical depth of the wildcard match. Power is reported as if all matched nodes had been explicitly named. Thus the maximum port depth in this scenario is the limit value +1.

If the `-by_node` is also the name of a port, its power is reported, as well as that of any connected subcircuit ports further down in the circuit hierarchy. It does not report any ports connected up in the hierarchy.

The `-except_port` argument excludes some ports from being reported. The proper way to conceptualize this behavior is to first determine which ports match the patterns specified by `-port` or `-by_node` and exclude any ports that match the pattern. For more details about excluding ports, see the following examples.

## Examples

The following example shows sample report file content.

```
##### LABEL=x1_ports_0_1u FROM=0 TO=1e-06 #####
Port Name: x1.vdd (vdd)
Sub-circuit Definition: buf8
Max(A) = 4.3845644e-03 at = 2.0847267e-07 Min(A) = 4.8172598e-12
at = 5.1890557e-09 Avg(A) = 1.2179682e-04 Rms(A) = 6.3463403e-04
Max(W) = 1.3153693e-02 at = 2.0847267e-07 Min(W) = 1.4451779e-11
at = 5.1890557e-09 Avg(W) = 3.6539046e-04 Rms(W) = 1.9039021e-03

Port Name: x1.x_buf1.vdd (vdd)
Sub-circuit Definition: buf4
Max(A) = 4.3099853e-03 at = 3.0768178e-07 Min(A) = 2.3878731e-12
at = 5.1890557e-09 Avg(A) = 5.9134919e-05 Rms(A) = 4.2502434e-04
Max(W) = 1.2929956e-02 at = 3.0768178e-07 Min(W) = 7.1636193e-12
at = 5.1890557e-09 Avg(W) = 1.7740476e-04 Rms(W) = 1.2750730e-03
```

`report_power` supports on-the-fly reporting. At the conclusion of a report window, as specified by the `-from` and `to` arguments, the report is written to the output file.

**Chapter 2: CustomSim Batch Command Syntax***Example 18 Reporting by Port Name*

```
.global vdd vss

.subckt level1 a
R1 vdd a r=1k
R2 a vss r=1k
X2 a level2
.ends

.subckt level2 a
R1 vdd a r=1k
R2 a vss r=1k
X3 a level3
.ends

.subckt level3 a
R1 vdd a r=1k
R2 a vss r=1k
X4 a level4
.ends

.subckt level4 a
R1 vdd a r=1k
R2 a vss r=1k
.ends

*Top level netlist start here, level=0
vvdd vdd 0 dc=3
vvss vss 0 dc=0
v0 a 0 dc=2
X1 a level1

.tran 1n 5n
.opt xa_cmd="set_sim_level 7"

*-limit is used to specify a hierarchical level:
.opt xa_cmd="report_power -port a -label atlevel0 -limit 0"
*The above does not report anything because 'a' is not a port,
*but is a net at top level (level 0).

.opt xa_cmd="report_power -port a -label atlevel1 -limit 1"
*The above reports power in X1.a, because a is a port for X1
*which is the 1st level of hierarchy (-limit 1)

.opt xa_cmd="report_power -port a -label sublevel2 -subckt level2"
*The above reports power in a port named "a" within subckt
*level2: X1.X2.a
```

```

.opt xa_cmd="report_power -port a -label porta"
*The above reports power thru all ports "a" in all subcircuits
*down to the hierarchical level 3 because if -limit is unspecified
*it defaults to 3:
*X1.A, X1.X2.A, X1.X2.X3.A; but not X1.X2.X3.X4.a

.opt xa_cmd="report_power -port vdd"
*The above report power thru all ports VDD in all subcircuits
*down to hierarchical level 3.

.end

```

*Example 19 Reporting by Connectivity*

```

.global vdd
vvdd vdd 0 3

X1 in out vdd 0 ckt1

.subckt ckt1 in out vdd vss
X1 in n1 vdd vss block1
X2 n1 n2 vdd vss block2
X3 n2 out block3
M1 vdd n1 n5 vss nmos
R1 n5 vss
.ends

.subckt block1 clkin clkout vdd vss
X1 clkin int1 vdd vss inv
X2 int1 clkout vdd vss inv
.ends

.subckt block2 input output vcc vss
X1 clkin int1 vcc vss inv
X2 int1 clkout vcc vss inv
.ends

.subckt block3 clkin clkout
X1 clkin int1 vdd vss inv
X2 int1 clkout vdd vss inv
.ends

```

If you specify the following command:

```
report_power -by_node vdd -limit 1
```

For the previous netlist, report\_power generates a power/current report of the following ports:

vvdd

x1.vdd

If you specify the following command:

```
report_power -by_node vdd -limit 2
```

This command generates a power/current report of the following ports:

vdd

x1.vdd

x1.x1.vdd

x1.x2.vcc

x1.x2.vdd

x1.x3.vdd

**Note:** Because of the global vdd statement exists in all subcircuit instances, but since the vdd port of the x1.x2 subcircuit is not connected, the report for this subcircuit is 0. Also, power/current report is generated for subcircuit and ideal voltage and current sources only. A report is not generated for other primitive elements or macro-models connected to the specified node, for example X1.M1.

If you specify the following command:

```
report_power -by_node vdd -limit 3
```

For the previous netlist, report\_power generates a power/current report of the following ports:

vvdd

x1.vdd

x1.x1.vdd

x1.x1.x1.vdd

x1.x1.x2.vdd

x1.x2.vcc

x1.x2.vdd

x1.x2.x1.vcc

x1.x2.x1.vdd

x1.x2.x2.vcc

```
x1.x2.x2.vdd  
x1.x3.vdd  
x1.x3.x1.vdd  
x1.x3.xd.vdd
```

**Note:** A power/current report is not generated for non-subcircuit elements or macro-models, for example, X1.M1.

To further illustrate port reporting, here is another sample netlist:

```
X1 nodeA out ckt1  
.subckt ckt1 in out  
X2 nodeB out ckt2  
..  
.ends  
.subckt ckt2 in out  
X3 nodeC out VBIAS  
..  
.ends  
.subckt VBIAS node1 VBIASOUT  
X4 node1 VBIASOUT VREG  
X5 VBIASOUT node2 LOAD  
  
.ends  
.subckt VREG IN OUT  
..  
.ends  
.subckt LOAD IN OUT  
..  
.ends
```

If you specify the following command:

```
report_power -by_node X1.out -limit 3
```

For the previous netlist, report\_power generates a power/current report of the following ports:

```
X1.out  
X1.X2.out  
X1.X2.X3.VBIASOUT
```

If you specify the following command:

```
report_power -by_node X1.X2.X3.VBIASOUT -limit 2
```

For the previous netlist, `report_power` generates a power/current report of the following ports:

X1.X2.X3.X4.OUT  
X1.X2.X3.X5.IN  
X1.X2.X3.VBIASOUT

**Note:** Although VBIASOUT is at level 3 of the hierarchy and `-limit 2` is specified the reports are still generated because the node is explicitly named.

Here is another example netlist with port reporting examples:

```
.subckt bottom a b
R1 a 0 r=1k
R2 b 0 r=1k
.ends

.subckt top 1 2
Rtop1 1 z r=100
Rtop2 2 y r=100
Xbottom 1 y bottom
.ends

Xtop a b top
Xtop2 a b top
```

If you specify the following command:

```
report_power -by_node xstop1
```

For the previous netlist, `report_power` generates a power/current report as follows:

- Node `xstop.1` is a port so it is reported. This port connects the node up the hierarchy.
- For `Xtop.xbottom.a`, this port is connected to the node `xstop.1`, but is down the hierarchy. Note that if you did not use the `-by_node` argument this port is not reported.
- This example does not report `xtop2.1`.

*Example 20 Sample Netlist Reporting with the -except\_port Argument*

```
.subckt superhuge3 vdd en
...
.ends
.subckt huge1 vdd en
...
.ends
.subckt huge2 vdd en
...
.ends
.subckt regulator vdd regvdd
...
.ends
.subckt tlc din dout vdd
Xreg vdd regvdd regulator
X1huge regvdd en1 huge1
X2huge regvdd en2 huge2
X3huge regvdd en3 superhuge3
.ends
Xchip din dout vdd tlc
Vvdd vdd 0 pwl 0 0 100n 3.3
```

If you specify the following command:

```
report_power -port vdd -except_port xchip.x3huge*
```

For the previous netlist, report\_power first finds all the ports that match vdd:

```
Xchip.vdd
Xchip.xreg.vdd
Xchip.x1huge.vdd
Xchip.x2huge.vdd
Xchip.x3huge.vdd
```

Then report\_power removes anything from the above that matches xchip.x3huge.\* , leaving:

```
Xchip.vdd
Xchip.xreg.vdd
Xchip.x1huge.vdd
Xchip.x2huge.vdd
```

If you specify the following command:

**Chapter 2: CustomSim Batch Command Syntax**

```
report_power -by_node xchip.regvdd -except_port  
xchip.x3huge*
```

For the previous netlist, report\_power first finds all the ports connected to xchip.regvdd:

```
Xchip.xreg.regvdd  
Xchip.x1huge.vdd  
Xchip.x2huge.vdd  
Xchip.x3huge.vdd
```

Then report\_power removes anything from the above that matches xchip.x3huge.\* , leaving:

```
Xchip.xreg.regvdd  
Xchip.x1huge.vdd  
Xchip.x2huge.vdd
```

Here is another sample netlist example to illustrate the -except\_port argument.

```
.subckt regulator vdd vdd3v3 vdd5v vaa vcc vee vss vss3v3 vss5v  
..  
.ends  
.subckt huge vdd vdd3v3 vdd5v vaa vcc vee vss vss3v3 vss5v  
..  
.ends  
.subckt tlc  
Xreg vdd vdd3v3 vdd5v vaa vcc vee vss vss3v3 vss5v regulator  
X1 vdd vdd3v3 vdd5v vaa vcc vee vss vss3v3 vss5v huge  
X3 vdd vdd3v3 vdd5v vaa vcc vee vss vss3v3 vss5v huge  
.ends  
Xchip tlc
```

If you specify the following command:

```
report_power -port v* -except_port *vee *vss* xchip.x3*
```

For the previous netlist, report\_power first finds all the ports that match v\*:

```
Xchip.xreg.vdd  
Xchip.Xreg.vdd3v3  
Xchip.Xreg.vdd5v  
Xchip.Xreg.vaa  
Xchip.Xreg.vcc
```

Xchip.Xreg.vee  
Xchip.Xreg.vss  
Xchip.Xreg.vss3v3  
Xchip.Xreg.vss5v  
Xchip.x1.vdd  
chip.X1.vdd3v3  
Xchip.X1.vdd5v  
Xchip.X1.vaa  
Xchip.X1.vcc  
Xchip.X1.vee  
Xchip.X1.vss  
Xchip.X1.vss3v3  
Xchip.X1.vss5v  
Xchip.x3.vdd  
Xchip.X3.vdd3v3  
Xchip.X3.vdd5v  
Xchip.X3.vaa  
Xchip.X3.vcc  
chip.X3.vee  
chip.X3.vss  
Xchip.X3.vss3v3  
Xchip.X3.vss5v

Then `report_power` excludes ports that match `*vee *vss* xchipx3*`, leaving:

Xchip.xreg.vdd  
Xchip.Xreg.vdd3v3  
Xchip.Xreg.vdd5v  
Xchip.Xreg.vaa  
chip.Xreg.vcc

## **Feedback**

### **Chapter 2: CustomSim Batch Command Syntax**

chip.x1.vdd  
Xchip.X1.vdd3v3  
Xchip.X1.vdd5v  
Xchip.X1.vaa  
Xchip.X1.vcc

*Example 21 Sample Netlist Reporting Using Wildcards*

```
.subckt fee r
rr r 0 r=1k
.ends

.subckt bar p q
r1 p 0 r=10
iq q 0 dc=0.1m
fee p fee
.ends

.subckt foo a b c
ra a a1 r=10
a a1 0 dc=1u
b b b1 r=10
ib b1 0 dc=2u
rc c c1 r=10
ic c1 0 dc=0.5u
xbar a b bar
xbar2 a1 b bar
.ends

.subckt top 1 2 3
x1 1 1 1 foo
x2 2 3 3 foo
x3 3 3 3 foo
.ends

xcut 1 2 3 top

v1 1 0 dc=1
v2 2 0 dc=2
v3 3 0 dc=3

.opt xa_cmd="report_power -by_node xcut.x1.* -label wildcardlimit
-limit 1"
* This reports nothing because all matched patterns exceed limit
value

.opt xa_cmd="report_power -by_node xcut.x1.a* -label
wildcard_a*_limit -limit 1"
* This reports nothing because all matched patterns exceed limit
value

.opt xa_cmd="report_power -by_node xcut.x1.a* -label
wildcard_a*_limit2 -limit 2"
```

```
* xcut.x1.a
* xcut.x1.ia (power_element)
* xcut.x1.xbar.p
* xcut.x1.xbar2.p

.opt xa_cmd="report_power -by_node xcut.x1.a -label
defined_a_limit2 -limit 2"
.opt xa_cmd="report_power -by_node xcut.x1.a1 -label
defined_a1_limit2 -limit 2"
* These two command together should produce the same output as
the previous

.opt xa_cmd="report_power -by_node xcut.1 -label defined_1_limit2
-limit 2"
.opt xa_cmd="report_power -by_node xcut.1 -label defined_1_limit1
-limit 1"
* because a named node always report the power of a port connected
to it, the above
* produces the same result.

.opt xa_cmd="report_power a -label port_a_limit2 -limit 2"
* note that when specifying by port, limit is a hard limit to the
port hierarchical level
* but with -by_node you can get 1 level deeper.
```

**See Also**

[set\\_wildcard\\_rule](#)

---

## report\_sim\_activity

Identifies performance problems in a simulation.

**Syntax**

```
report_sim_activity -type report_type
  [-node node_name {node_name}]
  [-except_node node_name]
  [-num max_nodes]
  [-twindow tstart [{tstop}] {tstart [tstop]}]
  [-file file_name]
  [-flush interval[%]]
  [-subckt subcircuit_name]
  [-except_subckt node_name]
  [-inst subcircuit_name]
  [-except_inst node_name]
```

```
[-probe_format format]
[-profile 0|1]
[-limit limit_value]
```

Argument	Description
<code>-type report_type</code>	Specifies one of the following report types: <ul style="list-style-type: none"> <li>▪ <code>node</code> reports the most highly active nodes. This option is the default if you do not specify <code>-type</code>.</li> <li>▪ <code>partition</code> reports only the most active partition information, which also includes the most active nodes in the partition.</li> <li>▪ <code>all</code> reports both the most highly active nodes and the most active partition information.</li> </ul>
<code>-node node_name {node_name}</code>	Defines the signal node name, which can be the node name of a single node or a node name with the asterisk (*) wildcard character that represents a group of node names. The behavior of asterisk (*) character is controlled by <a href="#">setWildcardRule</a> .
<code>-except_node node_name</code>	Specifies the nodes to be excluded from reporting. You can use a wildcard character in the node name.
<code>-num max_nodes</code>	Specifies the maximum number of nodes to report. The default is 10.
<code>-twindow tstart tstop {tstart tstop}</code>	Performs the check within the time window defined by <code>tstart tstop {tstart tstop}</code> . The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified it is assumed to be the end of the simulation. The final <code>tstop</code> can also be the <code>end</code> or <code>END</code> keyword. If you specify more than one window in <code>-twindow</code> , the windows must be in ascending order.
<code>-file file_name</code>	Specifies the name of the file that contains the activity information. The default file name is <code>prefix.cpa_time</code> .

Argument	Description
<code>-flush <i>interval</i>[%]</code>	Specifies when the information is flushed to the activity report. If the value is followed by the percent sign (%), the information is flushed every percentage of the transient run time. If the value is not followed by the percent sign, then the information is flushed every "wall time" period. This period is specified in hours. Decimal numbers are allowed.
<code>-subckt <i>subcircuit_name</i></code>	The activity report applies only to the specified subcircuit. You can specify multiple subcircuit names and use wildcard characters in a subcircuit name.
<code>-except_subckt <i>subcircuit_name</i></code>	The activity report does not apply to the specified subcircuit. Note that you cannot use wildcard characters in an subcircuit name in this case.
<code>-inst <i>inst_name</i></code>	The activity report applies only to the specified instance. You can specify multiple instance names and use wildcard characters in an instance name.
<code>-except_inst <i>inst_name</i></code>	The activity report does not apply to the specified instance. You can use wildcard characters in an instance name.
<code>-probe_format <i>format</i></code>	Specify one of the following options to dump probe statements to the activity report. An <i>output_filename.probe</i> file is generated with a list of probes. <ul style="list-style-type: none"> <li>▪ <i>hspice</i> format is <code>.probe tran v()</code>.</li> <li>▪ <i>xa</i> format is <code>probe_waveform_voltage ...</code></li> </ul>
<code>-profile 0 1</code>	When set to 1, an instance-based cost estimation is dumped to a <code>.cpa</code> file. The default is 0. This option helps you identify CPU-consuming blocks. You can speed up a simulation by replacing these blocks with models.
<code>-limit <i>limit_value</i></code>	Limits the number of instances reported by <code>-profile</code> . The default value is 10.

## Description

If you specify both the `-twindow` and `-flush` arguments, the activity information is written only within the time window. If you use Ctrl+C to stop a simulation, the activity information is written in a separate file (with a `.cpa` extension).

If you specify multiple time windows and use Ctrl+C, the reported activity is from the beginning of the last time window until the time use Ctrl+C. For example, if you specify `-twindow 10u 20u 50u 60u`:

- Ctrl+C at 15u (in the time window) writes activity information from 10u to 15u to the `.cpa` file.
- Ctrl+C at 30u (out of the time window) writes activity information from 10u to 30u to the `.cpa` file.
- Ctrl+C at 70u (out of the time window) writes activity information from 50u to 70u is written to the `.cpa` file.

**Note:** Because this feature is based on statistical sampling, some randomness in the results is to be expected.

## Examples

```
report_sim_activity -type all -twindow 1e-3 1.5e-3 3e-3 5e-3 -  
file MyAnalysis
```

Analyzes performance in the 1ms to 1.5ms and 3ms to 5ms time windows and stores results in the `MyAnalysis.cpa_1.5m` and `MyAnalysis.cpa_5m` files.

```
report_sim_activity -type all -flush 0.1
```

Enables the performance analyzer and flushes results every 6 minutes (1/10th of an hour).

```
report_sim_activity -type node -subckt ldo_iso -file MyRpt -  
twindow 20m 40m 100m 110m
```

Stores performance analysis results in two files, `MyRpt.cpa_40m` and `MyRpt.cpa_110m`. They contain the activity report for the `ldo_iso` subcircuit only for the 20m to 40m and 100mto 110m time windows.

## set\_active\_net\_flow

Triggers the automated active net flow.

### Syntax

```
set_active_net_flow [-switch] switch_value
[-vtol numeric_value]
[-twindow tstart tstop {tstart tstop}]
[-reuse_active_net enable_value]
[-reuse_ic enable_value]
[-setup_cmd cmd_file]
```

Argument	Description
-switch <i>switch_value</i>	Runs the active net flow. The default is 1.
-vtol <i>numeric_value</i>	Specifies the tolerance voltage difference. A node is considered active if the node voltage varies larger than the specified value. The default value is 100mV.
-twindow <i>tstart tstop</i> { <i>tstart tstop</i> }	Specifies the time windows to be checked for active nodes. If you specify this argument, the check is limited to the specified time windows.  The default for <i>tstart</i> is 0 and <i>tstop</i> is the end of transient simulation time. The keywords end and END are supported for <i>tstop</i> to specify the end of the simulation. The time window has to be specified in a pair, except for the last entry. If the last entry has only one value, <i>tstop</i> defaults to END.
-reuse_active_net <i>enable_value</i>	Instructs the CustomSim tool to reuse the active net file information and avoid rerunning the first pass for a data sweep. The default is 1, which specifies to reuse the active net file. Set this option to 0 to rerun the entire flow and regenerate the active net file. This reuse of the active net file is also the default behavior for .alter/bisection optimization flow.
-reuse_ic <i>enable_value</i>	By default the CustomSim tool reuses the prelayout .nodeset. If you enable this option, the CustomSim tool uses .ic instead of .nodeset.

---

Argument	Description
-setup_cmd <i>cmd_file</i>	By default, the CustomSim tool uses the same command for both the first and second run. If you specify this option, you can change three commands in the setup run to further speed-up the setup simulation. The three commands are: <a href="#">set_sim_level</a> , <a href="#">set_model_level</a> , and <a href="#">set_synchronization_level</a> .

---

**Description**

The active net flow automatically runs a prelayout simulation (ignores the [load\\_ba\\_file](#) command) and generates the active net information to use in the postlayout simulation. Its usage is limited to the back-annotation flow.

**Examples**

```
set_active_net_flow 1 -vtol 100m
```

Invokes the active net flow. A net is active when its voltage variation is greater than 100mV.

```
set_active_net_flow 1 -vtol 100mv -twindow 1u 2u
```

Invokes the active net flow. A net is active when its voltage variation is greater than 100mV within the [1us, 2us] window.

```
set_active_net_flow 1 -vtol 0.1 -twindow 1u 2u 4e-6s 6e-6s  
-reuse_active_net 0
```

Invokes the active net flow and reruns the entire flow to regenerate the active nets. A net is active when its voltage variation is greater than 100mV within the [1us, 2us] window or [4us,6us] window.

---

## set\_analysis\_core

Assesses the CustomSim analysis module (AM) processing requirements and can allocate up to two additional cores.

**Syntax**

```
set_analysis_core -core [0|1|2]
```

Argument	Description
-core 0   1   2	Sets one of the following AM multicore options: <ul style="list-style-type: none"> <li>▪ 0 disables multicore processing so the AM uses the same number of cores as the simulation engine. This is the default.</li> <li>▪ 1 detects if an additional thread is needed for analysis processing based on the processing load. When there are many active analyses, the CustomSim tool might enable an additional core to speed up processing.</li> <li>▪ 2 detects if an additional thread is needed for analysis processing based on the processing load. When there are many active analyses, the CustomSim tool might enable up to two additional cores to speed up processing.</li> </ul>

**Description**

The AM performs many different types of analysis, such as signal probing, signal measuring, CircuitCheck, and so on. The AM detects the amount of processing needed and might enable additional cores to speed up processing according to the `-core` value you specify. You might want to change the default of 2 depending on the load of the machine running simulation analyses.

**Note:** The `set_analysis_core` command is not compatible with the `set_multi_core` command. If you specify both of these commands, the CustomSim tool issues a warning message.

---

## set\_array\_option

Provides user control over the effects of the CustomSim SPICE optimization for array (SOFA) technology.

**Syntax**

```
set_array_option -array_detection enable_value
[-cell_gndcap_tol tol_value]
[-cell_subckt cell_subckt_name[.controlling_port]]
[-flash_array enable_value]
[-rc_optimize enable_value]
```

Argument	Description
<code>-array_detection enable_value</code>	Determines whether the CustomSim tool applies array optimization. Specify 0 to turn array optimization off. The default is 1.
<code>-cell_gndcap_tol tol_value</code>	Sets the relative tolerance for equalizing capacitor values for isolated ground capacitors inside memory cells. Values of ground caps are modified by not more than $tol\_value$ ( $abs((c-c\_modified)/c) \leq tol\_value$ ) to make more in-cell capacitors equal to each other across the different cells forming an array. This helps array optimization for some post-layout memory arrays. The valid range is 0% to 100%. The default is 15%.
<code>-cell_subckt cell_subckt_name .controlling_port</code>	Lets you manually provide the memory cell subcircuit name with optional controlling ports. the CustomSim tool automatically detects the controlling port of the memory cell, but if the CustomSim tool failed to do that, you can add one or more controlling port names right after the <code>cell_subckt_name</code> with the <code>".</code> concatenation. Controlling ports generally represent word line ports of the memory subcircuit.
<code>-flash_array enable_value</code>	If set to 1, this option enables detection of arrays of flash cells of single MOSFET or floating gate. The default is 0.
<code>-rc_optimize enable_value</code>	Set this option to 1 (default) to apply post-layout array optimization (SOLE4SOFA). Set it to 0 to disable post-layout array optimization.

**Description**

The SOFA algorithm transforms a memory array cell to an optimized array model to minimize their solve times while maintaining accuracy from the current `set_sim_level` settings. `set_array_option` can be useful for debugging simulations, or in other special circumstances when it is desirable to have the ability to disable the array optimization.

---

**set\_ba\_option**

Adjusts how the CustomSim tool handles back-annotation.

## Syntax

```
set_ba_option [-short_pins switch_value]
[-lump_c_only switch_value]
[-dpf_scale scale_value]
[-dpf_elem_type type {type}]
[-finger_prefix prefix_string]
[-active_net_file file_name {file_name}]
[-report_large_net value]
[-enable_error_net setting_value]
[-keep_prelayout_cap switch_value]
[-bus_delimiter left_char righ_char]
[-select_ipin_method method_value]
[-swap_port port1 port2 sub_name]
[-min_res value]
[-max_res value]
[-min_cap value]
[-report_trim_rlc enable_value]
[-rcnet net_name {net_name}]
[-cnet net_name {net_name}]
[-ccnet net_name {net_name}]
[-skipnet net_name {net_name}]
[-dpf switch_value]
[-skip_powernet mode_value]
[-skip_signalnet switch_value]
[-spftlv enable_value]
[-position_file file]
[-keep_prelayout_model switch_value]
[-auto_map_ba_terminal [0|1]]
[-cc_scale scale_value]
[-ccap_to_gcap value]
```

---

Argument	Description
-short_pins <i>switch_value</i>	Sets one of the following values: <ul style="list-style-type: none"> <li>▪ 0 off no false to disable shorting multiple pins together from the DSPF/SPEF file.</li> <li>▪ 1 on yes true to short multiple pins together (the default).</li> </ul>

Argument	Description
<code>-dpf_scale <i>scale_value</i></code>	Applies a <i>scale_value</i> to the DPF section of the SPF file. This is a global setting and applies to all SPF files that are used for a given simulation. The default value for <i>scale_value</i> is 1, meaning no scale applies to the DPF section and the CustomSim tool assumes the actual proper dimension is used in the DPF section. This argument takes both a unit or numerical value such as <code>1u</code> or <code>1e-6</code> .
<code>-lump_c_only <i>switch_value</i></code>	When set to 1, reads in the net capacitance value from the NET line and only back-annotate the lumped capacitance of that net and ignore the distributed RC back-annotation. The default is 0.
<code>-dpf_elem_type <i>type</i> {<i>type</i>}</code>	Lets you specify the types of elements you want to back-annotate from a DPF file. <a href="#">Table 6</a> shows the supported element types. Note that <code>-dpf_elem_type mos</code> is equivalent to the <code>-dpf_mos_only 1</code> option in the previous CustomSim release.
<code>-finger_prefix <i>prefix_string</i></code>	Instructs the CustomSim tool to use the <i>prefix_string</i> to identify finger devices. The default string is the @ character.
<code>-active_net_file <i>file_name</i>{<i>file_name</i>}</code>	Instructs the CustomSim tool to back-annotate the full RC for the active nets from the SPF/SPEF file. For inactive nets, the lump C value is added. You must use this argument with the <a href="#">load_ba_file</a> command.
<code>-report_large_net <i>value</i></code>	Reports large SPF nets as warnings in the CustomSim log file. By default, any net with an RC element count greater than 1000 is reported in the CustomSim log file.

Argument	Description
<code>-enable_error_net setting_value</code>	<p>Specifies one of the following three options:</p> <ul style="list-style-type: none"> <li>▪ 0   off   no   false ignores back-annotation of the net (the default).</li> <li>▪ 1   on   yes   true ignores elements connected to incorrect terminal only. This argument tells the CustomSim tool to ignore the back-annotation of an instance (* I) with an incorrect terminal, and any element connected to it. All other elements are back-annotated to the net.</li> <li>▪ Lump back-annotes lumped capacitance to the net. This argument tells the CustomSim tool to add the total lumped capacitance specified in the back-annotation file to the net.</li> </ul> <p>If source (SRC) is misconnected to drain (DRN), or vice-versa, the CustomSim tool automatically swaps the connection of those two terminals. If any other misconnection exists, the simulator does not back-annotate the net unless the <code>-enable_error_net</code> option is used.</p> <p>You can use this option only once. Otherwise the last entry overrides all previous entries.</p>
<code>-keep_prelayout_cap switch_value</code>	<p>Determines how the CustomSim tool deals with a prelayout netlist that contains ground capacitors with the same node name as the SPF net in the DSPF file. Specify one of the following values:</p> <ul style="list-style-type: none"> <li>▪ 0 overwrites the prelayout net capacitance with what is defined in the DSPF netlist.</li> <li>▪ 1 (default) keeps all the ground capacitors in the prelayout netlist and add the remaining capacitors in the back-annotation file.</li> </ul> <p>This option applies only to ground capacitors. Coupling capacitors are not considered. Also, the CustomSim tool issues a warning message when the prelayout net capacitance is overwritten by the DSPF netlist.</p>
<code>-bus_delimiter left_char right_char</code>	<p>Modifies the bus character only inside the SPEF file. You can specify the <code>left_char</code> and <code>right_char</code> to match the prelayout node names. The supported bus delimiters are [] and &lt;&gt;.</p>

Argument	Description
<code>-select_ipin_method method_value</code>	Instructs the CustomSim tool to select which ipin to use for any type of analysis such as <code>.probe</code> / <code>.measure</code> / <code>.ic</code> / <code>.nodeset</code> , or any statement that uses the NET name. This option has no impact for NETs with a <code>*   P</code> statement.  You can specify one of the following values: <ul style="list-style-type: none"><li>▪ 0 to choose the first instance pin as the representative node. This is the default value.</li><li>▪ 1 to choose the last instance pin as the representative node.</li><li>▪ 2 to choose the first instance pin as the representative node after sorting of <code>(*   I)</code>.</li><li>▪ 3 to choose the last instance pin as the representative node after sorting of <code>(*   I)</code>.</li></ul>
<code>-swap_port port1 port2 sub_name</code>	Allows port swapping for subcircuit (macromodel) terminals.
<code>-min_res value</code>	Specifies the global lower threshold of the resistors from the back-annotated DSPF/SPEF to be kept. All resistors with an absolute value below the specified value are shorted.
<code>-max_res value</code>	Specifies the global upper threshold of the resistors from the back-annotated DSPF/SPEF to be kept. All resistors with an absolute value above the specified value are treated as open-circuit.
<code>-min_cap value</code>	Specifies the global lower threshold of the capacitors from back-annotated DSPF/SPEF to be kept. All capacitors with an absolute value below the specified value are treated as open-circuit.
<code>-report_trim_rlc enable_value</code>	Generates an <code>.rlcignore</code> file with the list of resistor, capacitor, and inductor elements that have been optimized. The default is 0.

Argument	Description
<code>-rcnet net_name {net_name}</code>	Specifies the nets to use for full RC back-annotation. All other nets not specified by this argument only use lumped capacitance back-annotation. The lumped capacitance value uses the net capacitance value in the * NET line. This argument works in both SPF and SPEF.
<code>-cnet net_name {net_name}</code>	Specifies for the nets to use lumped capacitance back-annotation. The lumped capacitance value uses the net capacitance value in the * NET line. All the nets not specified by this argument use full RC back-annotation (assuming the net itself contains RCs). This argument works in both SPF and SPEF formats and accepts wildcard characters.
<code>-ccnet net_name {net_name}</code>	Specifies for the nets to use Cg+Cc back-annotation (removes all resistors from a given net). All the nets not specified by this argument use full RC back-annotation (assuming the net itself contains RCs). This argument works in both SPF and SPEF formats and accepts wildcard characters.
<code>-skipnet net_name {net_name}</code>	Specifies the name of nets that should not be back-annotated. You can specify multiple net names. You can use wildcard characters in the net names.
<code>-dpf switch_value</code>	Specifies to back-annotate the DPF section. The default <i>switch_value</i> is 1.
<code>-skip_pownet mode_value</code>	Specify one of the following options for <i>mode_value</i> :
	<ul style="list-style-type: none"> <li>▪ <code>external</code> to disable back-annotation into external power nets (ground or connected to VSRC).</li> <li>▪ <code>internal</code> to disable back-annotation into internal power nets.</li> <li>▪ <code>all</code> to disable back-annotation into all power nets. The default is not to skip any power nets.</li> </ul>
<code>-skip_signalnet switch_value</code>	Specifies signal nets to be skipped for back-annotation.
<code>-spftlv enable_value</code>	Enables the top-level view of the SPF file for the reliability analysis (RA) flow. The default is 0.

Argument	Description
<code>-position_file file</code>	Reads in a position file when multiple SPF files are read in for RA.
<code>-keep_prelayout_model switch_value</code>	Sets which model the CustomSim tool uses, the model in the prelayout netlist or in SPF file (default). Usually, different models are used for prelayout netlist and post-layout netlist. When running simulation with a prelayout netlist and back-annotation with SPF extracted from layout, the value for this option can decide which model should be used.
<code>-auto_map_ba_terminal [0   1]</code>	Enables or disables automatic mapping for macro models terminal names used in the prelayout netlist to the pin names in the post-layout netlist. The default is 0.  For more information about this options, see the <a href="#">Mapping Terminals</a> section.
<code>-cc_scale scale_value</code>	Applies a <i>scale_value</i> to the parasitic coupling capacitors in the DSPF files. This is a global setting and applies to all DSPF files that are used for a given simulation. The default value is 1, meaning no scale applies to the parasitic coupling capacitors.
<code>-ccap_to_gcap value</code>	Specifies the global capacitance value of the coupling capacitors to be kept as a result of back-annotated DSPF/SPEF. This argument converts all the coupling capacitors to ground capacitors if the value of the coupling capacitor is below the specified <i>value</i> .

### Description

By default, the CustomSim tool uses [load\\_ba\\_file](#) for back-annotation. You can use [set\\_ba\\_option](#) when you want to make adjustments to how the CustomSim tool handles files to perform lumped capacitance back-annotation only, or when you want to short out the SPF pins or apply the DPF scaling to the SPF files.

**Note:** This command now has the functionality of the `enable_ba_error_net` command, which is now obsolete.

### ***Supported Element Types***

[Table 6](#) lists the elements you can specify with the `-dpf_elem_type` option.

*Table 6    Supported Element Types*

Supported Element Keywords	Expected To Match
mos	m* or M*
diode	d* or D*
resistor	r* or R*
bjt (BJT)	q* or Q*
capacitor	c* or C*
instance (Xxxx macro-model)	x* or X*

### ***Mapping Terminals***

When parsing the prelayout netlist, the CustomSim tool finds all macro models from the model libraries (.lib files) and picks up a device from the macro model definition that represents a macro model. For each macro model, the CustomSim tool provides subcircuit port-to-device pin mapping. This information is used to generate corresponding [map\\_ba\\_terminal](#) commands to map macro model ports to device pin names.

When you run the [load\\_ba\\_file](#) command, the `-auto_map_ba_terminal` option automatically generates a [map\\_ba\\_terminal](#) command for macro models that maps the terminal name used in the prelayout netlist to the pin name in the postlayout netlist. This option provides correct back-annotation and accurate simulation results. Also, for devices with interchangeable terminals such as MOSFETs, resistors, capacitors and inductors, a corresponding `set_ba_option -swap_port` command is automatically generated to define swappable device terminals. Without this functionality you need to provide mapping manually.

**Table 7** show the devices that are supported in macro model definitions with their pin names used for mapping.

*Table 7 Swappable Device Pins*

Device	Pin Name	Swappable Pins
MOSFET	D, G, S, B	D and S
JFET	D, G, S	D and S
BJT	C, B, E	No pin swapping
DIODE	A, C	No pin swapping
RES, CAP, IND	P, N	P and N

## Examples

Given the following vss net call:

```
* | NET vss 0.00924425PF
* | I (x04/mn:BULK x04/mn BULK B 0 20.15 8.55)
* | I (x02/mn:BULK x02/mn BULK B 0 22.75 1.05)
* | P (vss B 0 0.325 0.151)
* | P (vss_1 B 0 20.15 8.55)
* | P (vss_2 B 0 22.75 1.05)
```

The vss\_1 and vss\_2 pins are shorted to vss if set\_ba\_option -short\_pins 1 is applied.

```
set_ba_option -bus_delimiter < >
```

This command changes the bus delimiters in the SPEF file. For example:

```
xtop/xsram/xcol1/wl[0] becomes xtop/xsram/xcol1/wl<0>
xtop/xsram/xcol<1>/wl[0] becomes xtop/xsram/xcol<1>/wl<0>
xtop/xsram/xcol[1]/wl[0] becomes xtop/xsram/xcol<1>/wl<0>
set_ba_option -dpf_elem_type mos
```

Back-annotates only MOSFET DPF devices and ignores other devices in the DPF section.

```
set_ba_option -dpf_elem_type mos resistor diode
```

Back-annotates only MOSFET/resistor/capacitors DPF devices and ignores other devices in the DPF section.

```
set_ba_option -dpf_elem_type mos instance
```

Back-annotates only MOSFET and Xx instance DPF devices and ignores other devices in the DPF section.

```
schematic netlist:  
M1 a b c d NCH l=45e-9 w=180e-9
```

```
SPF:  
M1 a b c d NMOS l=45-9 w=90e-9 ...  
M1@2 a b c d NMOS l=45e-9 w=90e-9 ....
```

The default value for `-keep_prelayout_model` is 0, which means it uses the model from SPF file. When setting `-keep_prelayout_model` to 1, it uses the model from the schematic netlist.

---

## **set\_bus\_format**

Sets the bus delimiters.

### **Syntax**

```
set_bus_format -open open_delimiter  
[-close close_delimiter]
```

---

Argument	Description
<code>-open open_delimiter</code>	Specifies the opening delimiter character, for example, [, <, _, or nothing (no open delimiter).
<code>-close close_delimiter</code>	Specifies the close delimiter character, for example, ], >, _, or nothing (no close delimiter).

---

### **Description**

You can specify two types of bus delimiters: both open delimiter and close delimiter, or just the open delimiter. If you do not use a `set_bus_format` command, by default the CustomSim tool uses [ ] as bus delimiters, for example, a[0], a[1], a[2], and so on.

### **Examples**

```
set_bus_format -open < -close >
```

**Specifies a<0>, a<1>, a<2>, and so on.**

```
set_bus_format -open /
```

**Specifies a/0, a/1,a/2, and so on.**

```
set_bus_format -open -close ]
```

**Specifies a0], a1], a2], and so on.**

```
set_bus_format -open -close
```

**Specifies a0, a1, a2, and so on.**

```
set_bus_format -open #
```

**Specifies a#0, a#1, a#2, and so on.**

---

## **set\_capacitor\_option**

Lets you control capacitor handling at different phases of the simulation.

### **Syntax**

```
set_capacitor_option -rule 1|2  
    [-min value]  
    [-report enable_value]
```

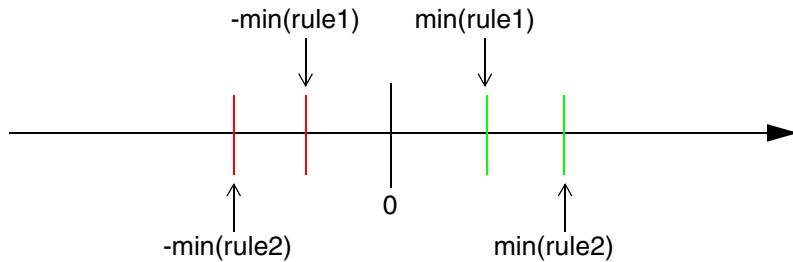
or

```
set_capacitor_option [-keep_negative_cap enable_value]  
    [-report enable_value]
```

Argument	Description
<code>-rule 1   2</code>	<p>Specifies one of the following optimization rules for capacitor elements:</p> <ul style="list-style-type: none"> <li>▪ Specify a value of 1 to process (open) the capacitors at the parsing stage, before the optimization and after resolving the parameters. This rule does not save elements in the database, which means you cannot recover them during a simulation. It has the advantage of using less memory.</li> <li>▪ Specify a value of 2 to save the capacitor elements in the database and process them during the optimization. Capacitors with absolute value below the specified value are lumped during optimization. This rule has the advantage of recovering capacitor elements for other analysis in the simulation. However, this type of analysis can impact the performance of the simulation.</li> </ul> <p>Note that you can specify both rules in one simulation, but only rule 1 can override rule 1, and rule 2 can override rule 2. Because rule 1 is executed before rule 2, make sure that you follow:  <math>\min(\text{rule1}) &lt; \min(\text{rule2})</math>. See <a href="#">Figure 6</a> for further details about specifying both rules in the same simulation.</p>
<code>-min value</code>	<p>Specifies the lower threshold value of the capacitors to be kept. All capacitors with an absolute value below the specified value are treated as follows by the <code>-rule</code> argument.</p> <ul style="list-style-type: none"> <li>▪ <code>-rule 1</code> specifies open.</li> <li>▪ <code>-rule 2</code> specifies lumped.</li> </ul>
<code>-report enable_value</code>	<p>Generates a <code>.rlcignore</code> file with the list of resistor, inductor, and capacitor elements that were optimized by <code>set_resistor_option</code>, <code>set_capacitor_option</code> and <code>set_inductor_option</code>. The default is 0.</p>
<code>-keep_negative_cap enable_value</code>	<p>Enables or disables the control for negative capacitors for a simulation. The default is 1 to keep negative capacitors (from both a netlist and back-annotation) for optimization by the <code>-rule</code> and <code>-min</code> arguments. If you want to disable (open) negative capacitors for simulation, you need to set this argument.</p>

## Description

Figure 4 shows the relationships between -rule 1 and -rule 2 in the same simulation.



$(\min \text{ of rule1}) < (\min \text{ of rule2})$  for positive capacitors  
 $-(\min \text{ of rule1}) > -(\min \text{ of rule2})$  for negative capacitors

Figure 4 Using -rule 1 and -rule 2 for capacitors in the same simulation

## See Also

[load\\_ba\\_file](#)  
[set\\_ba\\_option](#)  
[set\\_inductor\\_option](#)  
[set\\_resistor\\_option](#)

## set\_ccap\_level

Lets you override the default coupling capacitor tolerances.

### Syntax

```
set_ccap_level -level ccap_level
  [-inst inst_name {inst_name}]
  [-subckt subckt_name {subckt_name}]
```

Argument	Description
<code>-level ccap_level</code>	Sets the level of the coupling effect. The <i>ccap_level</i> is an integer from 1 to 7, where 1 produces the fastest simulation and 7 the most accurate. The default is the level you specify using <a href="#">set_sim_level</a> , or, if none was specified, level 3. See <a href="#">Table 8</a> for details about each level.
<code>-inst inst_name {inst_name}</code>	Defines the instance name at which the <i>ccap_level</i> applies.
<code>-subckt subckt_name {subckt_name }</code>	Defines the subcircuit name at which the <i>ccap_level</i> applies.

**Description**

During simulation, the CustomSim tool uses the default coupling capacitor tolerances pre-set in each [set\\_sim\\_level](#). You can use `set_ccap_level` to override the default coupling capacitor tolerances. You can use this command locally on specific instances or subcircuits.

[Table 8](#) provides guidelines for when to use `set_ccap_level` and what level to choose.

*Table 8 Guidelines for using set\_ccap\_level*

Level	Description
1	Specifies the most aggressive setting. All coupling capacitors are split. No coupling effects are modeled at this level.
2	Specifies an aggressive setting. All coupling capacitors, except those with both terminals that belong to the same resistive network, are split to ground. Almost no coupling effects are considered during simulation, in particular no inter-block coupling effects are modeled.  Note that you should not use this level when charge-coupling effects are important for circuit functionality, such as in charge-pump circuits.

*Table 8 Guidelines for using set\_ccap\_level*

<b>Level</b>	<b>Description</b>
3	Performs maximum optimizations of coupling capacitors. A small amount of coupling effects in blocks propagating digital signals and power distribution networks, and some coupling effects in blocks propagating analog signals are modeled. This level is suitable for functional verification or for simulation of only the strongest coupling effects in digital, memory, low-sensitivity analog, mixed-signal and full-chip circuits.
4	Performs moderate optimizations of coupling capacitors. It models strong coupling effects, particularly in blocks propagating analog signals. This level is suitable for functional verification or for simulation of only the strongest coupling effects in all circuits.
5	Performs balanced optimizations of coupling capacitors. Coupling effects in both digital and analog circuits are accurately modeled for all types of circuits. This level is suitable for accurate timing and power simulation for all circuits with coupling effects.
6	Performs conservative optimizations of coupling capacitors. Detailed coupling effects are accurately modeled to provide SPICE-like accuracy for all types of circuits. All circuits with coupling effects would generate SPICE-like accuracy.
7	No coupling capacitor optimizations are performed. This level yields full-SPICE accuracy.

**Note:** In all cases, the `set_ccap_level` corresponds to the matching level used by the `set_sim_level` command. That is, if you set `set_ccap_level` to 3, the results are the same as if you had `set set_sim_level` to 3.

### Examples

```
set_sim_level 3
set_ccap_level 6
```

In this example, the CustomSim tool overrides the coupling capacitor tolerances from 3 (set by `set_sim_level`) to level 6.

**See Also**[set\\_sim\\_level](#)

---

## set\_ccap\_option

Use this command with [set\\_ccap\\_level](#) to get maximum performance simulating designs with many small coupling capacitors.

**Syntax**

```
set_ccap_option -ccap_to_gcap ccap_threshold_value
    [-ccap_to_scap switch_value]
    [-ccap_threshold ccap_low_threshold_value]
```

Argument	Description
<code>-ccap_to_gcap</code> <code><i>ccap_threshold_value</i></code>	Converts all coupling capacitors to ground capacitors if the coupling capacitor value is smaller than the <code><i>ccap_threshold_value</i></code> . There is no default value for the <code><i>ccap_threshold_value</i></code> . You must specify a value to perform the conversion.
<code>-ccap_to_scap</code> <code><i>switch_value</i></code>	Converts all coupling capacitors to simplified capacitors. The simplified capacitor is an approximated model that allows partitioning across two regions previously connected by the coupling capacitor. The switch value can specify: <ul style="list-style-type: none"><li>▪ 1 for a more efficient way of handling the capacitor model.</li><li>▪ 2 for a more conservative way of handling the capacitor model.</li></ul>
<code>-ccap_threshold</code> <code><i>ccap_low_threshold_value</i></code>	Specifies to keep any coupling capacitors with values greater than the <code><i>ccap_low_threshold_value</i></code> value. The default is 1.e-10F.

## Description

The `set_ccap_level` command controls how the CustomSim tool handles the coupling capacitors. In general, this command provides a good performance/accuracy tradeoff. However, in designs with many small coupling capacitors, especially from a large back-annotation file, `set_ccap_level` might not provide enough tuning capability to get maximum performance. `set_ccap_option` provides more advanced controls.

The `set_ccap_option` command splits small coupling capacitors to ground capacitors before the `set_ccap_level` takes effect, so specify `set_ccap_option` before you use `set_ccap_level`. The `set_ccap_option` command converts small coupling capacitors prior to the RC optimization.

## Examples

```
set_ccap_option -ccap_to_gcap 1e-18
```

Splits all coupling capacitors to ground capacitors if the couple capacitor value is less than 1e-18.

```
set_ccap_option -ccap_to_scap 1
```

Splits all coupling capacitors to simplified capacitors.

---

## **set\_circuit\_flash**

Specifies a single, easy-to-use command to simulate flash designs.

### Syntax

```
set_circuit_flash -app func|power -type 0|1
```

---

Argument	Description
<code>-app func power</code>	Use <code>-app func</code> to check the functionality of a flash circuit. Use <code>-app power</code> for accurate measurement of power consumption.
<code>-type 0 1</code>	Specify <code>-type 0</code> to get good accuracy for a flash circuit with charge pumps and oscillators. Specify <code>-type 1</code> for a dynamic power simulation.

---

## Description

A flash design typically is a complex circuit with memory cells, charge pumps, voltage regulators and digital logic, each of which might operate at its own voltage range and have its own accuracy requirement. In addition, you may want to simulate a flash circuit in different flows such as ideal, post-layout extracted, and power-up/power-down. These complexities often make it difficult to come up with a set of commands that optimize accuracy and performance for the simulation.

To meet this challenge, the `set_circuit_flash` command has been created to enable internal customization according to the characteristics of the flash design, accuracy requirement, and type of the simulation.

For checking the functionality of a flash circuit, use `-app func`. For this application, the default value of the option is `-type 0`, which has good accuracy for a flash circuit with charge pumps and oscillators. If your circuit does not have these blocks, set `-type 1` for better performance.

For accurate measurements of power consumption, use `-app power`. For this application, the default value of the option is also `-type 0`. It gives you the highest accuracy, appropriate for leakage power measurements. For dynamic power simulation, the accuracy requirement is less stringent, so you can use `-type 1` for better performance.

This command is a global command. When using it, remove all other global commands that affect accuracy, such as `set_sim_level 5`. However, you can keep all other global commands such as `set_duplicate_rule -select_subckt last`, and all local commands such as `set_model_level 5 -subckt osc_23`.

## Examples

```
set_circuit_flash -app power -type 1
```

Specifies a dynamic power simulation.

```
set_circuit_flash -app func -type 0
```

Specifies a functional simulation for a flash circuit with charge pumps.

```
set_circuit_flash -app func -type 1  
set_model_level 5 -subckt chg_pmp
```

Specifies a functional simulation for a flash circuit with a known block that has a charge pump. Note that for this example you can also use the setting in the

previous example. However, the setting in this last example gives you better performance because it uses `-type 1`, so you have a local command for the charge pump to satisfy the high accuracy requirement for this block.

---

## set\_current\_option

Adds an extra control for the current accuracy in addition to the [set\\_sim\\_level](#) setting.

### Syntax

```
set_current_option [-level] option_value
```

---

Argument	Description
<code>option_value</code>	You can specify an integer value of 0-7. The higher the value, the more accurate the current measurement. A value of 1 or 2 is recommended for peak, average and rms current measurements. A value equal or greater than 3 is recommended for leakage current or sensitive measurement.  In general, the more the number of decades needed to settle from the preceding peak, the higher the option value should be. When tiny changes in voltages can influence the supply currents cumulatively, a value greater than 3 may be needed. For extremely detailed current measurements of value of 6 is recommended.

---

---

## set\_dc\_option

Controls DC convergence.

### Syntax

```
set_dc_option  
[-method method_type [method_type ...]]  
[-skip_dc value] [-cont_dc enable_value]  
[-min_res min_res_value]  
[-report report_type]  
[-iteration iter_value]
```

```

[-loop_solve loop_solve_val]
[-max_v max_val]
[-min_v min_val]
[-latch latch_value]
{-cck_filepath directory_path}

```

Argument	Description
-method <i>method_type</i>	<p>Specifies one of the following <i>method_type</i> options:</p> <p><i>auto</i> is the default sequence that the CustomSim tool uses to achieve DC convergence. The CustomSim tool starts with the most aggressive method and, if DC convergence is not achieved, goes to a more conservative method in the following order: static -&gt; adaptive -&gt; spice.</p> <p><i>static</i> partitions the circuit before solving the operating point. It uses the switch level for its initial estimate and the Newton Raphson iteration and pseudo-tran approaches for single-matrix solving.</p> <p><i>adaptive</i> automatically adjusts the partition during the solving stage for better DC convergence. When you specify this method, the block size is often larger than with the <i>static</i> method. It also uses a more robust Newton Raphson iteration and pseudo-tran approach for single-matrix solving.</p> <p><i>spice</i> is the most accurate DC method. It does not partition and uses the Newton Raphson iteration and pseudo-tran approaches for solving in one single matrix.</p> <p>For more information about these options, see the note in the Description section.</p>
-skip_dc <i>value</i>	<p>Specifies one of the following values:</p> <p>0 (default) uses the usual DC methods.</p> <p>1 specifies to skip the DC evaluation and go straight to transient analysis. Use this setting only for debugging the IC file from another simulator imported in the CustomSim tool for initial condition testing.</p>

Argument	Description
<code>-cont_dc enable_value</code>	<p>When DC does not converge, the CustomSim tool stops and sets all the unsolved nodes to 0V and continues with the transient simulation. This is the default behavior of CustomSim DC when <code>enable_value</code> is off.</p> <p>If <code>enable_value</code> is on, the CustomSim tool continues solving the rest of the nodes at the last algorithm chosen in the case of non-convergence and applies the initial condition appropriately to the nodes. For example, when <code>-method static -cont_dc 1</code> is specified and the CustomSim tool does not converge using the static method, it continues solving the rest of the nodes using the static method. And if <code>-method static adaptive -cont_dc 1</code> is used and the CustomSim tool does not converge in both methods, the CustomSim tool continues solving the rest of the nodes using adaptive method.</p> <p>See <a href="#">Common Syntax Definitions</a> for more about <code>enable_value</code>.</p>
<code>-min_res min_res_value</code>	<p>Without this argument, the DC solution keeps all the resistors when it solve the operating point. This arguments tells DC processing to short small resistors below the specified <code>min_res_value</code>. Note that this argument does not have a default value.</p> <p>This option only supports the <code>static</code> method of <code>-method</code>.</p>
<code>-report report_type</code>	<p>Specifies one of the following <code>report_type</code> options:</p> <ul style="list-style-type: none"> <li><code>always</code> specifies to always dump the DC convergence report.</li> <li><code>fail</code> specifies to only dump the DC convergence report when the DC operating point fails to converge (the default).</li> <li><code>off</code> does not dump the DC convergence report.</li> <li><code>unstable</code> reports the percentage of total iterations for each node during DC initialization. The report is dumped into the <code>.unstable</code> file.</li> </ul>
<code>-iteration iter_value</code>	Specifies the maximum number of DC iterations.

Argument	Description
<code>-loop_solve loop_solve_val</code>	If you set this value greater than or equal to 1, it controls the DC maximum number of region solving only in static analysis. The default is 400.
<code>-max_v max_val</code>	If you set this value greater than or equal to 0, it clamps voltage in DC analysis. The default value is 1000000.
<code>-min_v min_val</code>	If you set this value less than or equal to 0, it clamps voltage in DC analysis. The default value is -1000000.
<code>-latch latch_value</code>	Specify one of the following values. <ul style="list-style-type: none"> <li>▪ 0 to ignore front-end latch detection.</li> <li>▪ 1 for front-end latch detection based on group latch nodes, with one node of each group assigned a nodeset based on vsupply detection on this group. This is the default value.</li> <li>▪ 2 for front-end latch detection based on group latch nodes, with one node of each group assigned initial conditions (ic) based on vsupply detection on this group.</li> </ul>
<code>-cck_filepath directory_path</code>	Specifies the directory path to check when searching for the CircuitCheck (CCK) latch detection files, If you do not specify a path, the CustomSim tool does not load any CCK latch detection files.

### Description

This command controls options related to the DC solver. You can use it to output a DC convergence report to aid in debugging a DC convergence problem. The report is written to a .dc0 file in the same directory as the other simulator output files. The converged and non-converging nodes are listed in separate sections together with the node voltage, delta-v, delta-I, and the element contributing the largest current to the node.

You can also choose the DC method or skip DC analysis to go straight to transient analysis.

**Note:** If you specify multiple options for the `-method` argument, the CustomSim tool starts with the most aggressive DC method and proceeds to more conservative DC methods if necessary to

achieve DC convergence. The following guidelines apply to the -method options.

*Table 9 How To Choose a DC Convergence Method*

-method option	Guideline
auto	Specifies the method to let the CustomSim tool choose the DC convergence method automatically.
static	Specifies the method that provides the best performance for most circuits of any size.
adaptive	Specifies the method for medium sized analog or mixed-signal circuits.
spice	Specifies the most accurate DC method in the CustomSim tool. It is not recommended for large circuits because of its impact on performance.  If the spice method still does not provide DC convergence, the CustomSim tool issues a non-DC convergence warning and starts the transient analysis from the non-converged operating point derived from this option.

### Examples

```
set_dc_option -report always
```

Specifies to always dump the DC convergence report.

```
set_dc_option -method static
```

Specifies to use the static method for DC analysis.

---

## set\_duplicate\_rule

Lets you process multiple subcircuit definitions with the same name, multiple ports in a subcircuit with the same port name, or multiple model definitions with the same name.

## Syntax

```
set_duplicate_rule -select_subckt select_value
[-subckt sub_name {sub_name}]
[-select_model select_value]
[-model model_name {model_name}]
[-subckt_port enable_value]
```

---

Argument	Description
-select_subckt select_value	Specifies if the <i>select_value</i> of the first or last subcircuit definition is used when duplicate definitions exist. The default setting is 0, which raises an error if duplicate subcircuit definitions exist.
-subckt sub_name {sub_name}	Scopes the -select_subckt setting to only the named subcircuit definitions.
-select_model select_value	Specifies if the <i>select_value</i> of the first or last subcircuit definition is used when duplicate definitions exist. The default setting is 0, which raises an error if duplicate model definitions exist.
-model model_name {model_name}	Scopes the -select_model setting to only the named model definitions.
-subckt_port enable_value	If a subcircuit definition contains a duplicate port name, the CustomSim tool raises an error unless this option is enabled. If enabled, duplicate ports are shorted together.

---

## Description

This command controls the simulator behavior when duplicate model or subcircuit definitions are found in the netlist. It also controls the behavior when duplicate subcircuit port names exist in a subcircuit definition. A duplicate model or subcircuit definition raises an error unless this command is used. It can be used to specify if the first or last definition of a subcircuit or model is used and can be issued multiple times. The last command issued takes precedence. As with other the CustomSim tool specifications, a global command does not override a previously set local command.

Duplicate subcircuit port names are allowed by default in Eldo simulation mode. In other netlist formats, duplicate subcircuit port definitions result in an error unless you specify the -subckt\_port argument to enable duplicate ports. All

duplicate ports are electrically shorted.

### **Examples**

#### *Example 22 Global Setting*

```
set_duplicate_rule -select_subckt last
```

Specifies a global setting to select the last subcircuit.

#### *Example 23 Global Precedence Setting*

```
set_duplicate_rule -select_subckt last
set_duplicate_rule -select_subckt first
```

Overrides all previous `set_duplicate_rule` commands and uses the first subcircuit definition.

#### *Example 24 Global and Local Precedence Setting*

```
set_duplicate_rule -select_subckt first -subckt B
set_duplicate_rule -select_subckt last
```

A global setting does not override a previously set local setting. In this example, subcircuit B uses the first definition and the remaining duplicated subcircuits use the last definition.

#### *Example 25 Requesting Errors for Unexpected Duplicates*

```
set_duplicate_rule -select_subckt last -subckt A B C
```

If a global setting is made, the CustomSim tool does not raise any parsing errors for duplicated definitions. If only local settings are made the CustomSim tool raise a parsing error for any other duplicate definition. If the netlist contained:

**Chapter 2: CustomSim Batch Command Syntax**

```
.subckt A
*first a definition
...
.ends

.subckt A
*second A definition (expected)
...
.ends

.subckt D
...
.ends

.subckt D
* not intentional or expected
...
.ends
```

The CustomSim tool selects the last definition of subcircuit A, but raises an error because subcircuit D has duplicate definitions.

---

## **set\_flash\_option**

Sets options for the flash core cell.

### **Syntax**

```
set_flash_option -delvto value
[-inst inst_name {inst_name}] |
[-subckt subcircuit_name {subcircuit_name}]
```

---

<b>Argument</b>	<b>Description</b>
-delvto value	Specifies the change in threshold voltage.
-inst inst_name {inst_name}	Selects flash cells to be initialized. Wildcards can be used in the <i>inst_name</i> .
-subckt subcircuit_name {subcircuit_name}	Selects flash cells for initialization by subcircuit name.

---

### Description

This command specifies the initial change in threshold voltage to be applied to flash cells. If `-inst` or `-subckt` is not used to specify specific instances the command applies to all flash cells in the netlist. A `delvto` setting applied by this command overrides the instance parameter `delvto` for a flash cell.

### Examples

```
set_flash_option -delvto -0.5
```

Sets an initial vth change of -0.5V on all flash cells in the netlist.

```
set_flash_option -delvto 2 -inst x1.x2.x3.mcell
```

Sets an initial vth change of +2V on the `x1.x2.x3.mcell` flash cell.

```
set_flash_option -delvto -1.5 -subckt memword_16
```

Sets an initial vth change of -1.5V on all flash cells in all `memword_16` instances of the subcircuit.

---

## **set\_floating\_node**

Sets the voltage value of floating nodes at the DC operating point or throughout the transient simulation.

### Syntax

```
set_floating_node -val value [-type gate|all]
                   [-format ic|vsr] [-outfile enable_value]
```

---

Argument	Description
<code>-val value</code>	Specifies a voltage value applied to the floating nodes.
<code>-type gate   all</code>	Specifies one of the following options: <code>gate</code> – Sets floating gates. <code>all</code> – Sets all floating nodes, except floating bulks (default).

Argument	Description
-format ic   vsrc	Specifies one of the following options: ic – Initializes floating nodes with the <i>value</i> (default)  vsrc – Sets floating nodes to the <i>value</i> throughout the simulation. Using this option is equivalent to applying voltage sources of <i>value</i> to the floating nodes.
-outfile enable_value	Enables (1, on) or disables (0, off) creation of an output file. See <a href="#">Common Syntax Definitions</a> for more about the <i>enable_value</i> . The filename is the output filename specified with the -o option when invoking the CustomSim tool (or the input filename, if -o was not specified). The file extension is .fnode. If you create an output file and set -format ic, the CustomSim tool saves output data in the form of .ic statements; if -format vsrc, the CustomSim tool saves output data in the form of voltage sources. You can modify the output file to include it into the simulation.

### Description

Behavior of floating nodes can vary considerably with simulation algorithms and models. In a few cases, simulation results for a critical part of a circuit can depend on the voltage value at one or more floating nodes in the circuit. This can lead to hard-to-detect design issues and cause different results from different simulators. Forcing such nodes to a known value by means of `set_floating_node` can help isolate design problems.

**Note:** You cannot use this command with the `report_floating_node` command. If you specify both the `set_floating_node` and `report_floating_node` commands, then the last one is used.

### Examples

[Example 26](#) initializes all floating nodes at 0v in OP. No output file is generated.

*Example 26*

```
set_floating_node 0 -type all -format ic
```

[Example 27](#) sets all floating gates to 3.3v throughout the simulation. the CustomSim tool prints an output file with an extension .fnode0, containing data in the form of voltage sources.

*Example 27*

```
set_floating_node 3.3 -type gate -format vsrc -outfile 1
```

---

## **set\_hotspot\_option**

Controls which nodes are reported by [check\\_node\\_hotspot](#).

### **Syntax**

```
set_hotspot_option -factor value
```

---

Argument	Description
-factor value	Specifies the value used in the calculation to determine which nodes are reported by the <a href="#">check_node_hotspot</a> command. The default is 0.5.

---

### **Description**

Nodes specified by the [check\\_node\\_hotspot](#) command are not reported when the sum of the capacitive charging and discharging currents is less than the *value* multiplied by the sum of the node with the largest capacitive currents.

### **Examples**

```
check_node_hotspot -node *
set_hotspot_option -factor 0.3
```

These commands enable hot spot checks on all nodes, but only report the nodes for which the sum of the capacitive charging and discharging current is more than 0.3 multiplied by the sum of the node with the largest capacitive current. The following example shows the format of the output file.

Node Name	Cap (fF)	Toggle	Icin (uA)	Icout (uA)
VDD	3.03e+05	0	1.825	3.187
_GND	2.646e+04	0	2.047	2.159
SAMPLINGA	307.1	6	1.553	1.562
Total non-input nodes		:	791	
Total node toggles		:	392	
Total charging current		:	18.87uA	
Total discharging current		:	19.46uA	

---

## **set\_identification\_rule**

Sets front-end identification rules, which set simulation strategies to maximize performance with good accuracy.

### **Syntax**

```
set_identification_rule
  -type force_digital|digital|mixed|analog|force_analog
  [-detect_source_coupled_fet digital|mixed|analog]
  [-force_parasitic_diode digital|mixed|analog]
  [-detect_current_mirror digital|mixed|analog]
  [-detect_diode digital|mixed|analog]
  [-less_diode_connected enable_value]
  [instance_spec]
```

Argument	Description
-type force_digital digital  mixed analog force_analog	<p>Specifies a macro setting that forces all arguments of <code>set_identification_rule</code> to use the same setting type. However, if a specific argument is used and its value is different from that of the <code>-type</code> argument, the individual argument value takes precedence.</p> <p>You can specify the following types:</p> <ul style="list-style-type: none"><li>▪ <code>force_digital</code> forces the specified instance to be digital. You use this type with the <code>instance_spec</code> argument.</li><li>▪ <code>digital</code> specifies that all arguments under the <code>set_identification_rule</code> command are set to digital.</li><li>▪ <code>mixed</code> specifies that all arguments under the <code>set_identification_rule</code> command are set to mixed.</li><li>▪ <code>analog</code> specifies that all arguments under the <code>set_identification_rule</code> command are set to analog.</li><li>▪ <code>force_analog</code> forces the specified instances to be analog. You use this type with the <code>instance_spec</code> argument.</li></ul>

Argument	Description
<code>-detect_source_coupled_fet digital mixed analog</code>	<p>The CustomSim tool default is to detect source-coupled FET based on a set of rules. The detection of source-coupled FET impacts partitioning such that the fewer source-coupled FETs detected, the better the partitioning. This argument instructs the CustomSim tool to modify the source-coupled FET detection rules.</p> <p>You can specify the following FET rules:</p> <ul style="list-style-type: none"> <li>▪ <code>digital</code> skips source-coupled FET detection.</li> <li>▪ <code>mixed</code> relaxes the source-coupled FET detection requirement so there are less source-coupled FETs detected compared to the CustomSim tool default setting.</li> <li>▪ <code>analog</code> is the default setting. The CustomSim tool uses the more conservative source-coupled FET detection rules.</li> </ul>
<code>-force_parasitic_diode digital mixed analog</code>	<p>The CustomSim tool has a default set of detection rules to detect parasitic diodes. This argument instructs the CustomSim tool to modify the parasitic diode detection rules. Partitioning usually improves when more diodes are converted to parasitic diodes.</p> <p>You can specify the following parasitic diode rules:</p> <ul style="list-style-type: none"> <li>▪ <code>digital</code> converts all netlist diodes to parasitic diodes.</li> <li>▪ <code>mixed</code> converts all netlist diodes to parasitic diodes, except for diodes that are detected as analog diodes.</li> <li>▪ <code>analog</code> is the default setting. The CustomSim tool uses the more conservative parasitic diode detection rules.</li> </ul>

Argument	Description
<code>-detect_current_mirror digital mixed analog</code>	<p>The CustomSim tool has a default set of detection rules to detect current mirrors. This argument instructs the CustomSim tool to modify the current mirror detection rules. Partitioning usually improves when fewer current mirrors are detected.</p> <p>You can specify the following current mirror rules:</p> <ul style="list-style-type: none"> <li>▪ <code>digital</code> disables all current mirror detection rules.</li> <li>▪ <code>mixed</code> applies more aggressively current mirror detection rules.</li> <li>▪ <code>analog</code> is the default setting. The CustomSim tool uses the more conservative mirror detection rules.</li> </ul>
<code>-detect_diode digital mixed analog</code>	<p>The CustomSim tool has a default set of detection rules to detect diodes based on the characteristic of diode, rather than the model name in the netlist.</p> <p>You can specify the following diode rules:</p> <ul style="list-style-type: none"> <li>▪ <code>digital</code> applies more aggressive diode detection rules. If a netlist marked a device as a diode with a diode model name, regardless the diode has a diode characteristic or not, the CustomSim tool marks the device as diode.</li> <li>▪ <code>mixed</code> is the default setting. The CustomSim tool detects a diode when a device inherits a diode characteristic behavior.</li> <li>▪ <code>analog</code> specifies the same behavior as <code>-mixed</code>.</li> </ul>

Argument	Description
<code>-less_diode_connected enable_value</code>	The CustomSim tool has a default set of detection rules to detect diode-connected MOSFETs based on the characteristics of the MOSFETs. You can specify one of the following values: <ul style="list-style-type: none"> <li>▪ 0 specifies the default diode-connected MOSFET detection.</li> <li>▪ 1 relaxes the diode-connected MOSFET detection requirement so there are less diode-connected MOSFETs detected compared to the CustomSim tool default setting.</li> </ul>

### Description

While the default CustomSim rules and strategies work well most of the time, there are cases when they can impact performance. So it is necessary for the CustomSim tool to provide some additional user controls to allow modifications to the front-end identification rules to gain better performance.

[Figure 5](#) shows examples of the topologies that are relevant to the CustomSim rules and strategies: DC means diode-connected, CM means current-mirror, and SC means source-coupled. (Note that the actual identification rules implemented in the CustomSim tool are more complicated than those shown in [Figure 5](#).)

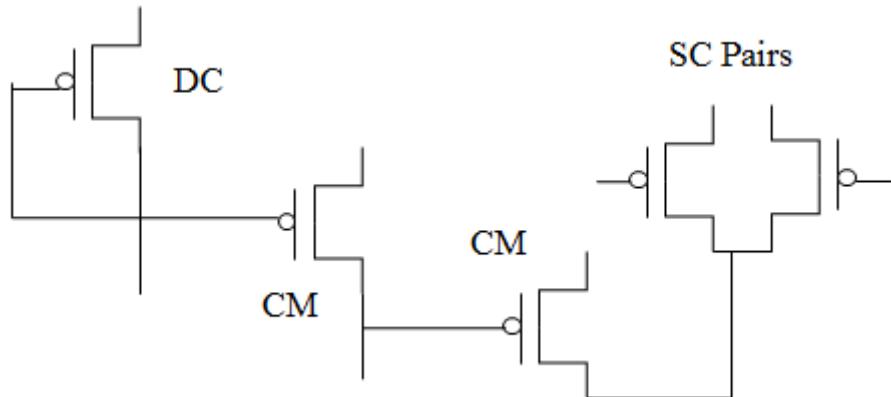


Figure 5 Simple Topologies Example

## Examples

```
set_identification_rule -detect_source_coupled_fet mixed
```

Applies the mixed-signal type of rules to source-coupled MOSFET detection so there are fewer source-coupled MOSFETs detected compared to the default detection rules.

```
set_identification_rule -force_parasitic_diode digital
```

Forces the diode identification rule such that all diodes are marked as parasitic diodes.

```
set_identification_rule -detect_current_mirror mixed
```

Applies the mixed-signal type of rules to current mirror detection so there are fewer current mirrors detected compared with the CustomSim default detection rules.

```
set_identification_rule -type digital
```

Applies the digital setting to all arguments

```
set_identification_rule -type digital -force_parasitic_diode analog
```

Applies the digital setting to all arguments but the -force\_parasitic\_diode argument uses the analog setting because the individual setting has higher precedence.

```
set_identification_rule -type force_digital -inst x1.x2
```

Forces the instances in x1.x2 to be identified as digital.

```
set_identification_rule -type force_digital -subckt inv
```

Forces the inv subcircuit to be identified as digital.

---

## set\_inductor\_option

Lets you control inductor handling at different phases of the simulation.

## Syntax

```
set_inductor_option -rule 1|2  
[-min value]  
[-report enable_value]
```

Argument	Description
-rule 1 2	Specifies one of the following optimization rules for inductor elements: <ul style="list-style-type: none"><li>▪ Specify a value of 1 to process (short) the inductor elements at the parsing stage, before the optimization and after resolving the parameters. This rule does not save elements in the database, which means you cannot recover them during a simulation. It has the advantage of using less memory.</li><li>▪ 2 to save the inductor elements in the database and process them during the optimization. Inductors with absolute values below the minimum value are shorted during optimization. This rule has the advantage of recovering inductor elements for other analysis in the simulation. However, this type of analysis can impact the performance of the simulation.</li></ul>
-min value	Specifies the lower threshold value of the inductors to be kept. All inductors below the specified value are shorted by -rule argument.
-report enable_value	Generates a .rlcignore file with the list of resistor, inductor, and capacitor elements that were optimized by set_resistor_option, set_capacitor_option and set_inductor_option. The default is 0.

## See Also

[set\\_capacitor\\_option](#)  
[set\\_resistor\\_option](#)

---

## set\_interactive\_stop

Enables the node voltage monitoring/checking capability.

## Syntax

```
set_interactive_stop -check v(node_name) {v(node_name)}
[-max vmax]
[-min vmin]
[-twindow tstart tstop {tstart tstop}]
[-cmf script_file_name]
```

or

```
set_interactive_stop -at interactive_mode_time
[-cmf script_file_name]
```

---

Argument	Description
-check v(node_name) {v(node_name)}	Specifies the voltage signals to be monitored during transient simulation.
-max vmax	If any monitored signal rises above the upper limit you specify within the specified time window, the CustomSim tool interrupts the transient simulation and enters interactive mode to facilitate debugging.
-min vmin	If any monitored signal falls below the lower limit you specify within the specified time window, the CustomSim tool interrupts the transient simulation and enters interactive mode to facilitate debugging.  Note that for monitored signals you should specify at least one -max or -min argument.
-twindow tstart tstop {tstart tstop}	Specifies the time periods for the signals to be monitored in the simulation.
-at interactive_mode_time	Specifies the time for the CustomSim tool to enter interactive mode.
-cmf script_file_name	Specifies the script file to be run when the CustomSim tool enters interactive mode due to a violation at the upper or lower limits of -max or -min.  Note that the commands in the script file must be interactive commands.

---

**Description**

If any monitored signal rises above the upper limit or falls below lower limit, the CustomSim tool interrupts the transient simulation and enters interactive mode. You can also use this command to specify a script file to run when entering interactive command due to either an upper/lower limit violation, or specified time with the -at argument. The secondary use of this command is to specify a time to re-enter interactive mode using the -at argument.

**Examples**

```
set_interactive_stop -at 150us
```

Stops the transient simulation and enters interactive mode at 150us.

```
set_interactive_stop -at 150us -cmf cmd_file
```

Stops the transient simulation and enters interactive mode at 150us and runs the commands in `cmd_file`.

```
set_interactive_stop -check v(net1) -max 1.2 -cmf cmd_file
```

Stops the transient simulation and enters interactive mode when `v(net1)` is greater than 1.2V and runs the commands in `cmd_file`.

---

**set\_latch\_control**

Detects latch nodes to prevent meta-stable states during power-up simulation.

**Syntax**

```
set_latch_control -twindow t0 t1
```

or

```
set_latch_control [-check 1]
```

---

Argument	Description
-twindow <code>t0 t1</code>	Specifies the power ramp-up period when force and release commands are applied to detected latch nodes.

---

Argument	Description
<code>-check enable_value</code>	Creates a <code>.metainfo</code> file with detected latch nodes, related VSRC sources, power node names and the suggested <code>set_latch_control</code> command to be used with the case.

**Description**

In a power-up simulation, the functional results or accuracy can fail due to meta-stable states of memory cells. Latch nodes can enter meta-stable states and cause larger supply current or unexpected current fluctuation during power ramping up. When power ramp-up is done, all the latches in the memory cells should be initialized with proper states, such as 0v and VDD. If the latch cells enter meta-stable states, a larger current is drawn from the power and can lead to other performance issues.

You can use the `set_latch_control` command to prevent latch nodes from entering meta-stable states to avoid such problems. This command can report the latch nodes using `-check 1`, or automatically apply force/release commands to the latch nodes during transient simulation by specifying `-twindow t0 t1`. Currently the type of latch nodes detected by this command are mostly SRAM memory cell nodes. Some latches with back-to-back inverters used in data buffer or cache could also be detected.

**Examples**

```
set_latch_control -twindow 1.5m 1.7m
```

Latch node voltages are controlled during the power-up period from 1.5ms to 1.7ms.

```
set_latch_control -check 1
```

Generates a `.metainfo` file, which lists detected memory cell latch nodes that can potentially lead to meta-stable states during power-up simulation. Only one out of the two back-to-back latch nodes is reported in the `.metainfo` file. In addition to the latch nodes, related power nodes are also reported. When the detected power nodes are external PWL VSRC, or DC VSRC, their VSRC definitions are also reported.

If the power nodes are internal, only their names are reported.

The following example shows a `.metainfo` file generated by `set_latch_control -check 1`.

```
FDI metastable nodes stats:  
xsram.xcore7.xb7_d75_wl240_241.xi0.q  
xsram.xcore6.xb0_d19_wl22_23.xi0.q  
xsram.xcore1.xb0_d70_wl0_1.xi1.q  
xsram.xcore5.xb0_d65_wl12_13.xi1.q  
....  
xsram.xcore7.xb7_d28_wl248_249.xi0.q  
xsram.xcore6.xb0_d70_wl20_21.xi0.q  
Total forced latch nodes=32768  
Total released latch nodes=32768  
supply_node=vddx_rf(external)  
* vvddx_rf Vsrc, pwl=0,0 1.6e-07,0.945 [Time-dependent]  
supply_node=xram.vdd_core(internal)  
#cell_latch=32768 #noncell_latch=0  
2 supply nodes found driving 65536 metastable latch nodes  
Suggested command:  
set_latch_control -twindow 0 1.6e-07
```

In this example, vvddx\_rf is a PWL VSRC defined as vvddx\_rf vddx\_rf 0 pwl (0 0 1.6e-07 0.945). Therefore you can specify set\_latch\_control -twindow 0 1.6e-07 to prevent metastable states during 0n and 1.6e-07s.

---

## set\_logic\_threshold

Specifies a default logic high/low threshold value.

### Syntax

```
set_logic_threshold -loth low_threshold_value  
-hith high_threshold_value  
[-node node_names]  
[-event_type value]
```

---

Argument	Description
-loth <i>low_threshold_value</i>	Sets the logic low threshold value. The default unit is Volt.
-hith <i>high_threshold_value</i>	Sets the logic high threshold value. The default unit is Volt.

---

Argument	Description
<code>-node node_names</code>	Specifies the node names to apply the threshold value. If this argument is not used, the default is all nodes. A wildcard is supported. You can specify multiple nodes separated by spaces.
<code>-event_type value</code>	Determines when a node generates a digital event. When <code>value</code> is set to 0, a node generates a digital event when its voltage rise above, or falls below, the middle voltage, such as $0.5 * (\text{high\_voltage} + \text{low\_voltage})$ . This setting is suitable to simulate most circuits.  When <code>value</code> is set to 1 (the default), a node generates a digital event for a node when its voltage rises above <code>high_threshold_value</code> or falls below its <code>low_threshold_value</code> . With this setting, the digital event is generated later than the one with the 0 setting.

### Description

The node state is determined by the relationship between its voltage and the high/low logic threshold value:

- If a node voltage is equal to or larger than the `high_threshold_value`, its state is ONE.
- If the node voltage is between the `high_threshold_value` and `low_threshold_value`, its state is undefined (U-state).
- If a percentage sign (%) is applied to the `high_threshold_value` and `low_threshold_value` arguments, a percentage value of the high/low voltage is taken, rather than taking the absolute value.

When the `set_logic_threshold` command is not used, the default `high_threshold_value` and `low_threshold_value` are 70% and 30%, respectively.

---

## set\_meas\_dump

Specifies whether to dump the `.ma` file used by the Measure Analyzer tool. The default is 0.

The `output .ma` file includes information about measurements, such as the original netlist statement, file and line number, numerical values of parameters involved, hierarchical dependencies of measurements, and dependent signals.

**Syntax**

```
set_meas_dump [0|1]
```

---

## **set\_meas\_format**

Sets the `.measure` file output format. If this command is not used, the default `.measure` file output uses CustomSim formatting.

**Syntax**

```
set_meas_format -format hspice|xa -bisect_meas final|all
```

---

Argument	Description
-format hspice xa	Specify hspice for <code>*.mt</code> output format (for HSPICE formatting) or xa for <code>*.meas</code> output format (for CustomSim formatting).
-bisect_meas final all	final is the default and means the CustomSim tool only writes the <code>.measure</code> results of the bisection iteration (transient simulation) that meets the goal of the bisection process. all outputs the <code>.measure</code> results for each iteration of the bisection process. This is useful for debugging the bisection process/results.

**Examples**

```
set_meas_format -format hspice
```

---

## **set\_message\_option**

Controls the number of warning messages the CustomSim tool prints.

**Syntax**

```
set_message_option -limit lim_val
```

or

```
set_message_option -action warn|stop|exit
```

or

```
set_message_option -pattern pat {pat} -limit lim_val
```

or

```
set_message_option -pattern pat {pat} -action warn|stop|exit
```

or

```
set_message_option -pattern pat {pat} -limit lim_val  
-action warn|stop|exit
```

or

```
set_message_option -limit lim_val -action warn|stop|exit
```

Argument	Description
-limit <i>lim_val</i>	Limits the maximum number of error messages to print. The default is 10. When you specify 0 is for the message, no warning is printed in the log file.
-action warn exit stop	Specifies one of the following options: <ul style="list-style-type: none"> <li>▪ warn (the default) to output warnings and continue the simulation.</li> <li>▪ exit to exit the simulation if a warning message is encountered.</li> <li>▪ stop to pause the simulation and prompt to either continue or exit.</li> </ul> If you specify exit or stop, the action takes place on any occurrence of the warnings specified with -pattern or any warning if no -pattern is specified.
-pattern <i>pat</i> { <i>pat</i> }	If you specify a pattern, the settings of the current command instance apply only to the specified warning message. If the pattern contains white spaces, it needs to be enclosed with a grouping delimiter, double quotes ("") or braces ({}). You can specify multiple patterns.

### Description

If you specify only the -limit argument, the CustomSim tool prints the specified number of warning messages for every warning type. If you specify both the -limit and -pattern arguments, the CustomSim tool prints the

specified number of messages that match the specified pattern. By default, the messages that do not match the pattern are printed 10 times.

If you specify both the `-pattern` and `-action` arguments, the CustomSim tool applies the action when the specified pattern is matched in a warning message: either continue, exit, or pause the simulation.

**Important:** It is recommended that the `set_message_option` command be used inside the command file included with the `-c` command line option.

## Examples

In [Example 28](#), when the CustomSim tool finds an ignored option or command, it stops the simulation and exits.

*Example 28*

```
set_message_option -pattern "Option/Command ignored" -action exit
```

In [Example 29](#), for any warning that contains the "floating" or "not matched" patterns, the CustomSim tool stops and exits the simulation.

*Example 29*

```
set_message_option -pattern "floating" -action exit
set_message_option -pattern "not matched" -action exit
```

In [Example 10](#), the CustomSim tool reports up to 100 warnings for the "Option/Command ignored" pattern and stops and exits the simulation if any messages contains the "floating" pattern.

*Example 30*

```
set_message_option -limit 100 -pattern "Option/Command ignored"
set_message_option -pattern "Floating" -action exit
```

In [Example 31](#), the CustomSim tool stops the simulation if it finds the "forward biased" or "exceeding" patterns. It displays the following prompt:

Message with stop action has been met. Do you want to exit the simulation? [y|n]

*Example 31*

```
set_message_option -pattern "forward biased" "exceeding" -action stop
```

In [Example 32](#), the CustomSim tool tabulates and print at most 50 occurrences of warnings containing the pattern "unsupported" and then exit with an error.

*Example 32*

```
set_message_option -limit 50 -pattern "unsupported" -action exit.
```

## **set\_model\_level**

Overrides the automatic choice of table model or equation used for each [set\\_sim\\_level](#) command. This command provides the arguments to either choose the type of model used for other [set\\_sim\\_level](#) settings or override it with a specific type of model.

### Syntax

```
set_model_level -level model_level | -type model_type
[-force 0|1]
[-inst inst_name {inst_name}]
[-subckt subckt_name {subckt_name}]
```

Argument	Description
-level <i>model_level</i>	Specifies an integer value of 1 through 7, which corresponds to a <a href="#">set_sim_level</a> value.
-type <i>model_type</i>	<p>Overrides the automatic model choice with a specific model specified by type. The type can be <code>digital</code>, <code>analog</code>, or <code>equation</code>. If you specify a less accurate type of model than the automatic model choice, the automatic model choice has precedence over your specification (unless you specify <code>-force 1</code>).</p> <ul style="list-style-type: none"> <li>▪ <code>digital</code> sets both the MOSFET current and charge model to fast table.</li> <li>▪ <code>analog</code> sets both the MOSFET current and charge model to table.</li> <li>▪ <code>equation</code> sets both the MOSFET current and charge model to equation.</li> </ul> <p>You cannot use both the <code>-level</code> and <code>-type</code> arguments in the same command.</p>

Argument	Description
<code>-force 0   1</code>	This option is only valid when you use <code>-type</code> . If you specify <code>-force 1</code> , CustomSim forces all MOSFETs to use the specific type of model specified with <code>-type</code> .
<code>-inst instance_name {instance_name}</code>	Specifies instance names.
<code>-subckt subcircuit_name {subcircuit_name}</code>	Specifies subcircuit names.

### Description

By default, XA chooses the type of table model or equation used for each `set_sim_level` command. `set_model_level` overrides the automatic choice of table model or equation used for each `set_sim_level`. It either chooses the type of model used for other `set_sim_level` commands or overrides it with a specific type of model.

### Examples

```
set_sim_level 3  
set_model_level 5
```

Overrides the model strategy of level 3 with level 5. All other strategies remain as level 3.

```
set_sim_level 4  
set_model_level -type analog
```

Overrides the model strategy of level 4 to promote digital table to analog (dynamic) table. If some MOSFETs use equation model, they are simulated using equation model.

```
set_sim_level 4  
set_model_level -type analog -force 1
```

Overrides the model strategy of level 4 to force all the MOSFETs to use analog (dynamic) table.

## set\_model\_option

Controls the type of model, the enhanced table feature, the BSIM4 NQS feature, the grid, MOS binning, and model parameter checking.

### Syntax

```
set_model_option model_spec
    [-grid value]
    [-grid_scale scale]
    [instance_spec]
```

where

```
model_spec ::= current_spec | charge_spec | stress_param_spec |
    b4nqs_model | binning_spec | model_param_check_spec
instance_spec ::= [-inst inst_name {inst_name}]
    [-subckt subckt_name {subckt_name}]
```

where

```
current_spec ::= -current current_model | -i current_model
charge_spec ::= -charge charge_model | -q charge_model
stress_param_spec ::= -lod lod_value | -wpe wpe_value
b4nqs_model ::= -b4nqs b4nqs_value
binning_spec ::= -mos_bin_ratio tolerance
    -mos_bin_closest model_name
model_param_check_spec ::= -model_param_check check_value
```

where

```
current_model ::= full | table | ids | fast | none
charge_model ::= full | table | fast | lump | none
```

Argument	Description
-current	Specify one of the following keywords to override the automatic choice for the current model:
<i>current_model</i> -i	
<i>current_model</i>	<ul style="list-style-type: none"> <li>▪ full for a MOS equation current model with no optimization.</li> <li>▪ table for a dynamic table built for each current in the MOS model.</li> <li>▪ ids for a dynamic table built only for drain-source current (ids) for a MOS model.</li> <li>▪ fast for a static table used for the current in a MOS model.</li> </ul>

Argument	Description
-charge <i>charge_model-q</i> <i>charge_model</i>	Specify one of the following keywords to override the automatic choice for the charge model: <ul style="list-style-type: none"> <li>▪ full for a MOS equation charge model with no optimization.</li> <li>▪ table for a dynamic table built for all the charge components in the MOS model.</li> <li>▪ fast for a static table used for all the charge components in a MOS model.</li> <li>▪ lump for approximate lump capacitors used for a MOS: Cgate, Csource, and Cdrain.</li> </ul>
-b4nqs <i>b4nqs_value</i>	Disables or ignores (0) b4nqs effects. Specify 1 (default) to invoke the b4nqs model.
-wpe -lod -0   off	Disables or ignores STI/LOD (for -lod) or WPE (for -wpe) effects.
-wpe -lod 1   on	Uses look-up tables, obtained from device equations, to handle stress effects. This is the default. See the Description section below for more details.
-wpe -lod 2	Uses an enhanced table model method, which uses a single base look-up table, to handle stress effects. See the Description section below for more details.

Argument	Description
<code>-mos_bin_ratio tolerance</code>	Specifies the ratio tolerance for binning the MOS model. It is an integer value between 0 and 500 and must always be followed by a percent sign (%). The default is 0.
	Post-layout simulations use extracted netlists with final sizes for devices. The final sizes extracted from layout are slightly different from the schematic sizes, so devices with the same size in prelayout (or schematic) netlist have slightly different sizes in the post-layout netlist. The difference is same but results in many devices each having its own size parameters. As a consequence, the CustomSim tool generates a large number of unique models, which can increase the memory usage and run time.
	The purpose of this option is to create fewer models and improve the memory usage and run time for post-layout simulations. All devices with parameters that differ by less than a given ratio share the same model. All parameters are assumed to have the same importance and the same ratio is used for all parameters.
<code>mos_bin_closest model_name</code>	Enables the use of the closest bin for a device that does not fit any of the given model bins. This feature applies to all device models specified by the <code>model_name</code> , where only the designated models have the feature applied, and all other models do not. You can specify a wildcard character ("**") to apply this feature to all models.  For more information about this option, see the <a href="#">Using mos_bin_closest</a> section.
<code>-grid grid_value</code>	Sets the absolute LUT grid size in volts. You can only specify this option globally.
<code>-grid_scale scale</code>	Scales the dynamic table model grid size by the square of the specified <code>scale</code> value. Setting the scale too small can lead to a very small grid size, which can use up a lot of memory and slowdown overall simulation. You can only specify this option globally. The default value is 1.

Argument	Description
<code>-model_param_check check_value</code>	This option is equivalent to the HSPICE option <code>.option MODPARCHK</code> . Use this option to determine whether a simulation aborts when it encounters fatal errors in model side parameter checking. When set to 1 (default), the simulation aborts if it encounters a fatal error during parameter checking and reports the error. When set to 0, it checks limited model parameters and re-sets them to avoid fatal errors and the simulation runs to conclusion. Note that this option is only available for BSIM4 models.
<code>-inst instance_name {instance_name}</code>	Specifies the instance names.
<code>-subckt subcircuit_name {subcircuit_name}</code>	Specifies the subcircuit names.

### Description

The CustomSim tool automatically chooses the appropriate current and charge model. You can overwrite this automatic selection and take the following stress effects into consideration:

- Shallow trench isolation (STI), a compact way of separating transistors for submicron geometries.
- Length of oxide definition (LOD), a measure of transistor layout sizing, used to modify device characteristics due to submicron effects.
- Well proximity effect (WPE), a measure of device degradation due to closeness of the transistor well or body to the transistor itself.

By default, the CustomSim tool builds a unique model (either an equation level model or a look-up table model) for each unique transistor geometry. A different model is required for a transistor even if the dimensions are the same as another transistor, differing only in the STI/LOD/WPE value. In this way, the CustomSim tool preserves all STI/LOD/WPE effects.

The CustomSim tool enables turning off the BSIM4 NQS parameter, if TRNQSMOD=1, without modifying the model file, to see if performance is impacted by using this parameter.

The CustomSim tool can also use an enhanced table model method, applied for a given transistor dimension. This model uses derating factors to account for STI/LOD/WPE effects during simulation. The algorithm provides performance and memory capacity advantages with minimal loss of accuracy.

You set the `-lod` and `-wpe` values to `0` or `off` to ignore STI/LOD/WPE effects, to `1` or `on` (default) to use preserve all stress effects, and to `2` to use the enhanced table model method.

The CustomSim tool can also:

- Control binning of MOS devices whose parameters fall within a given percent ratio of one another.
- Automatically choose the appropriate grid and grid scale. You can overwrite this automatic selection.
- Abort or continue when fatal model parameter checking errors occur.

**Note:** This command replaces the `use_mos_instpar` and `set_mos_binning` commands.

### ***Using mos\_bin\_closest***

For a MOS device with width (W) and length (L) specified on the instance line, where W and/or L do not fit any of the given model bins, the closest bin is defined by the following, in which Lmin, Lmax, Wmin, and Wmax are the minimum L, maximum L, minimum W, and maximum W, respectively, of a bin as shown in [Table 10](#).

*Table 10 Definition of Closest Bin*

Condition	Closest Bin
No W bin, L bin exists	$L_{min} \leq L < L_{max}$ , if $W >$ maximum $W_{max}$ of $L_{min}, L_{max}$ bin else $W \leq$ minimum $W_{min}$ of $L_{min}, L_{max}$ bin
W bin, no L bin	$W_{min} \leq W < W_{max}$ , if $L >$ maximum $L_{max}$ of $W_{min}, W_{max}$ bin else $L \leq$ minimum $L_{min}$ of $W_{min}, W_{max}$ bin
No W bin, no L bin	If $W >$ maximum available $W_{max}$ and ( $L >$ maximum $L_{max}$ of maximum available $W_{max}$ bin or $L \leq$ minimum $L_{min}$ of maximum available $W_{max}$ bin) else $W \leq$ minimum available $W_{min}$ and ( $L \leq$ minimum $L_{min}$ of minimum available $W_{min}$ bin or $L >$ maximum $L_{max}$ of minimum available $W_{min}$ bin)

In the following example [Table 11](#) gives the available bins and [Table 12](#) illustrates the closest bin selection for a given device size:

*Table 11 Example Bins*

<b>Bin</b>	<b>Lmin</b>	<b>Lmax</b>	<b>Wmin</b>	<b>Wmax</b>
1	2u	3u	50u	60u
1	3u	4u	50u	60u
3	6u	7u	40u	80u
4	6u	7u	80u	90u

*Table 12 Closest Bin Selection Examples*

<b>Condition</b>	<b>Device L</b>	<b>Device W</b>	<b>Closest Bin</b>	<b>Reasoning</b>
No W bin, L bin exists	6u	100u	4	$L_{min} \leq L < L_{max}$ , $W >$ maximum Wmax of Lmin, Lmax bin
No W bin, L bin exists	6u	60u	3	$L_{min} \leq L < L_{max}$ , $W \leq$ minimum Wmin of Lmin, Lmax bin
W bin, no L bin	5u	50u	2	$W_{min} \leq W < W_{max}$ , $L >$ maximum Lmax of Wmin, Wmax bin
W bin, no L bin	1u	50u	1	$W_{min} \leq W < W_{max}$ , $L \leq$ minimum Lmin of Wmin, Wmax bin
No W bin, no L bin	5u	65u	2	If $W >$ maximum available Wmax and $L >$ maximum Lmax of maximum available Wmax bin

*Table 12 Closest Bin Selection Examples*

Condition	Device L	Device W	Closest Bin	Reasoning
No W bin, no L bin	1u	65u	1	If W > maximum available Wmax and L <= minimum Lmin of maximum available Wmax bin
No W bin, no L bin	1u	40u	1	W <= minimum available Wmin and L <= minimum Lmin of minimum available Wmin bin
No W bin, no L bin	5u	40u	2	W <= minimum available Wmin and L > maximum Lmax of minimum available Wmin bin

If you use `-mos_bin_closest` a warning appears in the log file that the MOS device with its W, L dimensions does not fit any given model bin. Also, the closest bin information (Lmin, Lmax, Wmin, Wmax) used for the respective MOS device is noted in the log file.

### Examples

The following example uses the full current model and full charge model.

```
set_model_option -i full -q full
```

The following example turns off the BSIM4 NQS parameter.

```
set_model_option -b4nqs 0
```

The following example uses the enhanced look-up table for all effects.

```
set_model_option -lod 2 -wpe 2
```

The following example uses the mos binning ratio feature set to 10%.

```
set_model_option -mos_bin_ratio 10%
```

The following example uses the grid and grid\_scale feature, set to 0.01V and 0.5, respectively.

```
set_model_option -grid 0.01
set_model_option -grid_scale 0.5
```

The following example uses the model parameter checking feature, set to 0 for the simulation to continue with .reset model parameters that are fatal.

```
set_model_option -model_param_check 0
```

The following enables the closest bin selection feature for all models.

```
set_model_option -mos_bin_closest *
```

The following example enables the closest bin selection feature for all moscap\* models only. All other models do not use the closest bin selection feature.

```
set_model_option -mos_bin_closest moscap*
```

---

## **set\_monte\_carlo\_option**

Specifies how to run traditional Monte Carlo analysis for transient simulation.

### **Syntax**

```
set_monte_carlo_option [-enable enable_value]
[-sample_output output_files]
[-parameter_file enable_value]
[-simulate_nominal enable_value]
[-dump_waveform enable_value]
```

---

<b>Argument</b>	<b>Description</b>
<code>-enable enable_value</code>	The default setting for this option is enabled and must be set to process monte carlo samples. If this option is disabled and the netlist contains a sweep monte statement, the sweep monte statement is ignored with a warning.

---

Argument	Description
<code>-sample_output output_files</code>	Specifies which Monte Carlo output files to keep: <ul style="list-style-type: none"> <li>▪ <code>first</code> keeps the output file from the first Monte Carlo run.</li> <li>▪ <code>all</code> keeps all the Monte Carlo output files from all runs.</li> <li>▪ <code>last</code> keeps the output file from last Monte Carlo run.</li> <li>▪ <code>none</code> specifies not to keep any output files. This is the default setting.</li> <li>▪ <code>n {n}</code> keeps the Monte Carlo output files from the specified runs.</li> </ul>
<code>-parameter_file enable_value</code>	If you enable this option it creates a <code>.mc_params</code> parameter file.
<code>-simulate_nominal enable_value</code>	By default, this option is enabled and runs the nominal, 0 sweep simulation. If you disable it, the CustomSim tool does not run the 0 sweep simulation and starts with the first sweep (1).
<code>-dump_waveform enable_value</code>	You must enable this option to dump waveforms. It dumps only the waveform files for those samples you specify with the <code>-sample_output</code> option.

### Description

You can run Monte Carlo analysis as an additional check to ensure that your design functions as expected. The CustomSim tool only supports traditional Monte Carlo analysis for transient simulations with the HSPICE netlist format. For more information about traditional Monte Carlo analysis, see the Monte Carlo - Traditional Flow and Statistical Analysis section in the *HSPICE User Guide: Simulation and Analysis*.

Traditional Monte Carlo defines a random variable with a distribution function. You can assign a random variable just as in HSPICE. The CustomSim tool supports the same distribution functions as HSPICE:

- `UNIF(nominal_val, rel_variation [, multiplier])`
- `AUNIF(nominal_val, abs_variation [, multiplier])`
- `GAUSS(nominal_val, rel_variation, num_sigmas [, multiplier])`
- `AGAUSS(nominal_val, abs_variation, num_sigmas [, multiplier])`
- `LIMIT(nominal_val, abs_variation)`

Whenever a parameter defined by one of these functions is assigned, a new unique random variable is generated. For more information distribution functions, see the .PARAMETER Distribution Function section in the *HSPICE User Guide: Simulation and Analysis*.

The .mc file contains a measurement summary of the Monte Carlo simulation. It is the primary output file and reports a summary of the statistical data for each measurement:

- nominal
- mean
- variance
- stddev
- avgdev
- min
- max
- median
- sample number of min
- sample number of max
- sample number of median
- 10 bin horizontal ascii histogram:
  - First column of the histogram is the bin center point.
  - Second column, nb, reports the number of samples in the bin.
  - Third column, freq, reports the percentage of samples in the bin.

*Table 13 Generated Waveform Files*

<b>-sample_output setting</b>	<b>-dump_waveform setting</b>	<b>Output File Format</b>	<b>Output Waveform File</b>
none	0	Any	No waveform file.
none	1	Any	No waveform file.
all	0	Any	No waveform file.

*Table 13 Generated Waveform Files*

<b>-sample_output setting</b>	<b>-dump_waveform setting</b>	<b>Output File Format</b>	<b>Output Waveform File</b>
all	1	Any	<i>xa.m0.format</i> <i>xa.m1.format</i> <i>xa.m2.format</i> ... <i>xa.mn.format</i>
0,10	0	Any	No waveform file.
0,10	1	Any	<i>xa.m0.format</i> <i>xa.m10.format</i>

*Table 14 Monte Carlo Output Files*

<b>File Name</b>	<b>Description</b>
.mc	Contains the Monte Carlo measurement summary of all measurements and an ASCII histogram of each measurement. Monte Carlo analysis produces this file when a .meas command is used in the netlist. This file is not generated if there is no measurement.
.mc.csv	Contains all of the data in the .mc file, except the histogram. You can read this .csv file in a spreadsheet program.

*Table 14 Monte Carlo Output Files*

File Name	Description
.meas/.mt	Contains all the Monte Carlo sample measurement results. Monte Carlo analysis always produces this file. It contains only the measure values, not the Monte Carlo parameter values.
	This file uses standard CustomSim measure formatting and has a .meas extension. If you specify HSPICE formatting in the <a href="#">set_meas_format</a> command, it has a .mt extension.
	When you specify <code>set_meas_format -format hspice</code> , the CustomSim tool must create a measure file that is syntactically compatible with HSPICE so that a waveview histogram can be viewed. When you choose the HSPICE measure format, the per sample and summary measure file has a .mt extension rather than the standard .meas extension.
.mc_params	The .mc_params file is a listing of all of the random variable values used in the simulation. It is equivalent to the HSPICE .mct0 file. The default file format is the CustomSim measure file format. You can choose an HSPICE file format with the <a href="#">set_meas_format</a> command.
Monte Carlo analysis generates the following output files on a per Monte Carlo sample basis if you specify <code>-sample_output</code> :	
.m# .wdf/fsdb/out	Waveform output file.
.m# .meas/mt	Measure file with a .meas extension when you specify the standard CustomSim measure format, or a .mt extension if you specify the HSPICE measure format with the <a href="#">set_meas_format</a> command.
.m# .ic	Initial conditions file (if requested).

## Examples

### *Example 33 Running Monte Carlo Analysis*

```
.tran 1n 100n sweep Monte=2
.opt xa_cmd="set_Monte_Carlo_option -sample_output all"
```

This example creates the following output files:

- `xa.log`
- `xa.mc`
- `xa.meas`
- Sample 0 files (nominal simulation)
  - `xa.m0.wdf`
  - `xa.m0.meas`
  - `xa.m0.ic`
- Sample 1 files (first random set)
  - `xa.m1.wdf`
  - `xa.m1.meas`
  - `xa.m1.ic`
- Sample 2 files (second random set)
  - `xa.m2.wdf`
  - `xa.m2.meas`
  - `xa.m2.ic`

*Example 34 .mc File Example*

```

* XA 64-bit LINUX D-2010.03-20100316 (built 17:04:04 Mar 16 2010)
* Build id: 1762407
* Copyright (C) 2010 Synopsys Inc. All rights reserved.

meas_variable = out
nominal = 0.000000e+00
mean = 8.228455e-05
varian = 5.832803e-08
stddev = 2.415120e-04
avgdev = 1.893783e-04
min = -1.568056e-04
max = 6.508503e-04
median = 3.212345e-04
run_min = 4
run_max = 8
run_median = 7

-7.603999e-05, nb = 2, freq = 2.000000e-01 | *****
4.725601e-06, nb = 4, freq = 4.000000e-01 | *****
8.549119e-05, nb = 0, freq = 0.000000e+00 |
1.662568e-04, nb = 2, freq = 2.000000e-01 | *****
2.470224e-04, nb = 0, freq = 0.000000e+00 |
3.277879e-04, nb = 1, freq = 1.000000e-01 | ****
4.085535e-04, nb = 0, freq = 0.000000e+00 |
4.893191e-04, nb = 0, freq = 0.000000e+00 |
5.700847e-04, nb = 0, freq = 0.000000e+00 |
6.508503e-04, nb = 1, freq = 1.000000e-01 | ****

```

The `.mc_csv` file in [Example 35](#) contains the same data as the `.mc` file, except for the ASCII histogram. It has CSV formatting. The first line is a header row:

*Example 35 .mc\_csv File*

```

meas_variable,nominal,mean,varian,stddev,avgdev,min,max, ...
out,0.000000e+00,8.228455e-05,5.832803e-08, ...

```

The `.meas` file in [Example 36](#) is the standard CustomSim measure file. It contains all the per sample measurement values. For example:

*Example 36 .meas File*

```
*  
* XA 32-bit LINUX D-2010.03-SP1-ENG2 (built 14:07:40 Jul 19 2010)  
* Build id: 1833524  
* Copyright (C) 2010 Synopsys Inc. All rights reserved.  
  
mc_index = 1  
  
rt      = 8.7888150e-12      targ = 1.5616071e-09      trig =  
1.5528183e-09  
dly     = 3.4741727e-10      targ = 5.7741727e-10      trig =  
2.3000000e-10  
temper  = 1.2500000e+02  
alter#  = 1.0000000e+00
```

## **set\_multi\_core**

Runs a multicore simulation.

### Syntax

```
set_multi_core [-core num_cores]  
[-check_model value]  
[-check_netlist value]
```

Argument	Description
<code>-core num_cores</code>	Specifies the number of cores to use with either a numerical value of 2 or more or the <code>max</code> keyword. The number of cores requested is automatically adjusted to the maximum available in the machine.  The <code>max</code> keyword specifies to use maximum number of cores available in the machine. Note that you need one CustomSim licence per each two cores.

Argument	Description
<code>-check_model value</code>	Lets you check for non-thread-safe and thread-safe models with one of the following values: <ul style="list-style-type: none"><li>▪ 0 does not perform the check (default).</li><li>▪ 1 checks to see if thread-safe models are internally tagged as such and causes the CustomSim tool to error out if the models are not tagged properly.</li><li>▪ 2 reports internally tagged non-thread-safe models with a warning.</li><li>▪ 3 combines 1 and 2 .</li></ul>
<code>-check_netlist value</code>	Helps you determine whether a netlist is a good candidate for a multicore simulation. If you enabled a multicore simulation already, this option is ignored. Otherwise one of following three messages is dumped in <code>*.log</code> file: <p>This netlist is likely to benefit from a multicore simulation.</p> <p>This netlist can possibly benefit from a multicore simulation.</p> <p>This netlist is unlikely to benefit from a multicore simulation.</p> <ul style="list-style-type: none"><li>▪ 0 does not perform the check (default).</li><li>▪ 1 checks the netlist, dumps the result in the log file, and continues the simulation.</li><li>▪ 2 checks the netlist, dumps the result in the log file, and stops the simulation.</li></ul>

---

**Examples**

```
set_multi_core -core 4
```

Runs a multicore simulation with four cores.

```
set_multi_core -check_netlist 2
```

Reports the multicore check results and stops the simulation.

```
set_multi_core -check_model 1
```

Checks if models are internally tagged as thread-safe and errors out if they are not tagged as thread-safe.

## set\_multi\_rate\_option

Enables the multi-rate technology mode.

### Syntax

```
set_multi_rate_option -mode 1|2
```

---

Argument	Description
-mode 1   2	The default is 2. The recommended setting of 2 enables the new multi-rate technology that can provide enhanced simulator performance and precision, particularly in digital designs. The value of 1 is available to maintain backward compatibility with the 2012.06 multi-rate algorithms.

---

## set\_oscillator

Applies the trapezoidal integration method to oscillator circuits.

### Syntax

```
set_oscillator -inst inst_name|-subckt subckt_name  
[-disable value]  
[-report value]
```

---

Argument	Description
-inst <i>inst_name</i>	Specifies an instance as an LC oscillator.
-subckt <i>subckt_name</i>	Specifies a subcircuit definition as an LC oscillator.
-disable <i>value</i>	Specifies a value of 1 to disable automatic detection of oscillator circuits. This option applies globally. You cannot use it to disable detection of instances or subcircuits. The default is 0.
-report <i>value</i>	Specifies a value of 1 to output additional log information. When automatic detection is on, all detected elements are listed in the log. The default is 0.

---

## Description

Trapezoidal integration is often the best integration method for LC oscillator circuits but can apply to other types of oscillators as well. If the characteristic frequency on an inductive network is between 0.5Hz and 800 MHz, an oscillator circuit is often identified automatically. Because the capacitors in an LC oscillator have a critical impact on the oscillation frequency, this command causes conservative treatment of the capacitors in the oscillator.

If the CustomSim tool fails to identify an oscillator automatically, you can use the `set_oscillator` command to manually specify the trapezoidal integration method. Trapezoidal integration is also applied to the regions (partitions) that contain the designated oscillators. You can also use this command for other circuit elements that require trapezoidal integration.

In a hierarchical netlist, apply trapezoidal integration to the inductor of the oscillator or the lowest-level subcircuit that contains the oscillator. In a flat netlist, apply it to the inductor.

**Note:** If you do not specify `-subckt` or `-inst`, the CustomSim tool issues a warning and does not apply trapezoidal integration to any instance. If automatic detection is enabled, the CustomSim tool still applies trapezoidal integration to any detected oscillators.

## Examples

The following example applies the trapezoidal integration method to an inductor instance.

*Example 37*

```
set_oscillator -inst x1.xosc.L1
```

The following example applies the trapezoidal integration method to an oscillator instance.

*Example 38*

```
set_oscillator -inst x1.xxtal
```

The following example applies the trapezoidal integration method to all instances of a subcircuit.

*Example 39*

```
set_oscillator -subckt crystal
```

The following example disables automatic detection of LC oscillator circuits.

*Example 40*

```
set_oscillator -disable 1
```

The following example prints additional log information.

*Example 41*

```
set_oscillator -report 1
```

The following example disables automatic detection and applies the trapezoidal integration method to the specified instance.

*Example 42*

```
set_oscillator -inst xxtal -disable 1
```

---

## **set\_parameter\_value**

Lets the CustomSim tool change the value of a specified parameter for selected instances without modifying the netlist. This command supports only the `delvto` parameter for MOSFET instances.

### **Syntax**

```
set_parameter_value -name delvto  
    -value delvto_value  
    instance_spec
```

---

Argument	Description
<code>-value delvto_value</code>	Specifies the <code>delvto</code> parameter value.
<code>instance_spec</code>	See the <a href="#">Common Syntax Definitions</a> section for details about the <code>instance_spec</code> argument.

---

### **Examples**

```
set_parameter_value -name delvto -value 0.5 x1.*
```

Changes the `delvto` value for all MOSFET devices inside `x1` to 0.5.

## set\_partition\_option

Overrides the automatic setting for circuit partitioning.

### Syntax

```
set_partition_option -sp enable_value
    [-ap enable_value]
    [-print_power enable_value]
    [ [-rule] gate|channel|never]
    [-inst inst_name {inst_name}]
    [-subckt subckt_name {subckt_name}]
    [-footer_switch 0|1]
```

Argument	Description
-sp enable_value	Turns static partitioning on (1, the default) or off (0). Static partitioning is not instance specific and does not affect the idealized power-net block region.
-ap enable_value	Turns advanced partitioning on (1, the default) or off (0). Advanced partitioning is not instance specific.
-print_power enable_value	Writes a list of power cut nodes to the .cut file.
-rule gate never	Allows partitioning of MOSFETs at the gate (high-impedance) terminal. The node touching the gate of the MOSFET is in a different partition than the source/drain terminals. If it is set to a Verilog-A instance, it allows partitioning of the ports of Verilog-A instance. The nodes touching the high-impedance ports of the Verilog-A instance are in different partitions, but the Verilog-A instance is still in one partition. Note that this option only allows the device to be partitioned, but does not guarantee it. The partitioning rules of surrounding devices can cause the device to remain un-partitioned in spite of the gate rule being applied to it. Specify channel to allow partitioning of MOSFETs at the channel (low-impedance) terminal.  never turns off the partitioning. The nodes touching the MOSFET are in one partition.

---

Argument	Description
<code>-inst inst_name {inst_name}</code>	Applies the partition rule defined by <code>-rule</code> to the specified instances.
<code>-subckt subckt_name {subckt_name }</code>	Applies the partitioning rule defined by <code>-rule</code> to the specified subcircuits.
<code>-footer_switch 0 1</code>	If your circuit has footer switches use the <code>-footer_switch 1</code> option to ensure that the footer switch is properly recognized. The default is 0.

---

**Description**

You can use the `set_partition_option` command to enable/disable the CustomSim tool partitioning algorithms. The CustomSim tool has two stages of partitioning: static partitioning and advanced partitioning. The static partitioning algorithm uses mature circuit heuristics to perform partitioning. The advanced partitioning algorithm uses more aggressive partitioning methods to try to further reduce partition sizes.

If `-sp 0` is set all partitioning is disabled and the `-ap` setting is ignored. You can disable the advanced partitioning with `-ap 0` to isolate partitioning issues or to assist in tightening simulation accuracy at a cost of simulation run time. Disabling static partitioning disables all circuit partitioning in the CustomSim tool. This is not recommended for large circuits because of the prohibitive simulation runtime.

**Examples**

```
set_partition_option -ap 0
```

---

## set\_postlayout\_meas

Checks all the back-annotated signals used in the `.meas` statements and reports the earliest or latest signals.

**Syntax**

```
set_postlayout_meas mname
  -trig [early|late] | [min|max]
  -targ [early|late] | [min|max]
```

Argument	Description
<i>mname</i>	Specifies the name of the measurement as specified in the .meas statement. You can use wildcard characters in the name.
-trig early late	<p>When you specify -trig early, the CustomSim tool picks the first signal reaching val=&lt;value&gt; among all the SPF signals of the trig net within the time window defined by the .meas statement.</p> <p>When you specify -trig late, the CustomSim tool picks the last signal reaching val=&lt;value&gt; among all the SPF signals of the trig net within the time window defined by the .meas statement.</p> <p>You must specify either the -trig or -targ argument.</p>
-trig min max	<p>When you specify -trig min, the CustomSim tool picks the smallest of all rise/fall times among all the SPF signals of the trig net within the time window defined by the .meas statement.</p> <p>When you specify -trig max, the CustomSim tool picks the largest of all rise/fall times among all the SPF signals of the trig net within the time window defined by the .meas statement.</p>
-targ early late	<p>When you specify -targ early, the CustomSim tool picks the first signal reaching val=&lt;value&gt; among all the SPF signals of the targ net within the time window defined by the .meas statement.</p> <p>When you specify -targ late, the CustomSim tool picks the last signal reaching val=&lt;value&gt; among all the SPF signals of the targ net within the time window defined by the .meas command.</p>

Argument	Description
-targ min max	<p>When you specify -targ min, the CustomSim tool picks the smallest of all rise/fall times among all the SPF signals of the targ net within the time window defined by the .meas statement.</p> <p>When you specify -targ max, the CustomSim tool picks the largest of all rise/fall times among all the SPF signals of the targ net within the time window defined by the .meas statement.</p>

### Description

The `set_postlayout_meas` command is a postlayout processing command. It applies only to CustomSim simulations with SPF back-annotation using the `load_ba_file` command. It checks all the back-annotated signals used in the `.meas` statements and reports the earliest or latest signals, depending on how you specify the `-trig` and `-targ` arguments. When you only specify `-trig`, the same signal (same device port) is used for the `targ` signal. When you only specify `-targ`, the same signal (same device port) is used for `trig` signal.

The `set_postlayout_meas` command picks either the first arriving signal among all net signals or the last arriving signal among all net signals. In the case of rise/fall time measurement, `trig` and `targ` are the same signal.

In addition, the following rules apply to `set_postlayout_meas`.

- The measurements affected by `set_postlayout_meas` must be at the top level. They are not honored if the measurements are embedded in subcircuits.
- Only `.measure` (delay, rise and fall) is supported. The `set_postlayout_meas` does not support any other `.measure` commands. Standard HSPICE `.MEAS` syntax is supported.
- The back-annotation net instance pins (lines starting with \* | I in the SPF file) are automatically dumped in the waveform file when `set_postlayout_meas` is used. Only \* | I pins are considered. The \* | S and \* | P pins are ignored.
- You cannot apply multiple `set_postlayout_meas` commands to the same `.meas` statement. If you want to apply different time windows or different trig/targ values, then you must define multiple `.meas` statements. If multiple `set_postlayout_meas` commands apply to the same measurement name, then a warning is issued and the last one is used.

***Other Affected Commands***

If you use the `set_postlayout_meas` command together with the `meas_post` command, and if you save the postlayout signals with `probe_waveform_voltage -ba_net`, the `set_postlayout_meas` command applies to the `.meas` commands. For example, suppose you run a simulation with `probe_waveform_voltage -ba_net A*` and after the simulation completes you wan to perform measurements. You can write a SPICE file with the measurements:

```
.meas tran Delay1 trig v(A1) val=1 rise=1 targ v(A2) fall=1 val=1  
.meas tran Delay2 trig v(B1) val=1 rise=2 targ v(B2) fall=2 val=1
```

And write a configuration file with:

```
meas_post -waveform xa.fsdb  
set_postlayout_meas Delay1 -trig early -targ late  
set_postlayout_meas Delay2 -trig early -targ late
```

Both the `delay1` and `delay2` measurements are honored, but only the first one (`Delay1`) uses the `set_postlayout_meas` options, because only the `A*` back-annotated nets are saved. The `B1` and `B2` back-annotated signal nets are not saved with `probe_waveform_voltage -ba_net`, so the `Delay2` measurement is done as usual. The `set_postlayout_meas Delay2` command is ignored.

The CustomSim tool supports the Eldo `.EXTRACT` commands and generates a `.meas` file identical to the `.meas` file generated by the HSPICE `.meas` command. So the `set_postlayout_meas` command applies to the results of the `.EXTRACT` measurement. For example, if the following Eldo command is in the netlist:

```
.EXTRACT label=dell tpddd(v(in), v(out), vth=1
```

After the simulation completes, a `.meas` file is created with the delay between `v(in)` falling and `v(out)` falling with `val=1V`. If you specify the following configuration file command:

```
set_postlayout_meas dell -trig early -targ late
```

The measurement uses the early slope for the `trig` signal (`v(in)`) and the late slope for the `targ` signal (`v(out)`).

**Examples**

In the case of rise/fall (also called slope or transition) time measurement, `trig` and `targ` are the same signal, so the `-trig` and `-targ` arguments are optional. The `-trig` and `-targ` values can be `min` or `max`. For example, the

following .meas statement measures the fall time (it's fall because trig value is greater than targ value) of signal SUM0.

```
.meas tran fall1 trig v(SUM0) val='0.9*vdd' fall=1 from=1n to=2n
targ v(SUM0) val='0.1*vdd' fall=1 from=1n to=2n
```

And uses this DSPF file:

```
* | NET SUM0 0.0134922PF
* | P (SUM0 O 1e-14 -459.55 80.95)
* | I (x0/x33/M2:SRC x0/x33/M2 SRC B 0 -462.5 36.25)
* | I (x0/x33/M1:SRC x0/x33/M1 SRC B 0 -462.5 11)
```

The CustomSim tool measures all fall times for SUM0, that is:

Fall time for x0/x33/M2 : SRC from 0.9vdd to 0.1vdd.

Fall time for x0/x33/M1 : SRC from 0.9vdd to 0.1vdd.

The CustomSim tool then compares all the results. If set\_postlayout\_meas -trig max is used, then the measurement reports the max value. If set\_postlayout\_meas -trig min is used, then the measurement reports the min value.

The following .meas statement in a netlist applies to a delay measurement:

```
.meas tran D1 trig v(X1.z) val=vdd/2 rise=1 from= 1n to= 2n targ
v(X2.c) fall=2 val=vdd/2 late from= 3n to= 4n
```

And the configuration file contains:

```
set_postlayout_meas D1 -trig early -targ late
```

To determine the triggering signal, the CustomSim tool checks all SPF signals related to X1.z. It picks the first rising signal reaching vdd/2 between 1ns and 2ns. To determine the target signal, the CustomSim tool checks all SPF signals related to X2.c. It picks the last falling signal reaching vdd/2 between 3ns and 4ns. Note that the CustomSim tool ignores the first occurrences of X2.c signals falling, it picks the second occurrence of all X2.c signals falling (because of fall=2).

The .meas0 file shows:

```
...
D1 = 1.8657638e-09 targ = 1.8657638e-09 trig = 0.0000000e+00
targ_signal=x2.c::x1.x3.mn2:gate
trig_signal=x1.z::x1.x3.mn1:drain
Temper = 2.5000000e+01
alter# = 1.0000000e+00
```

## set\_powernet\_level

Controls the performance/accuracy tradeoff for IR drop simulation.

### Syntax

```
set_powernet_level -level level_value
```

---

Argument	Description
-level level_value	Specifies the performance/accuracy tradeoff level for IR drop simulation. The default is the level you specify with the <a href="#">set_sim_level</a> command. Levels 3 through 7 are valid. If you do not specify a <a href="#">set_sim_level</a> command, the default value is 3.

---

### Description

The higher the `set_powernet_level`, the more parasitic elements are taken into consideration for IR drop simulation. If the resistor connected to the power supply is below the predetermined value for each [set\\_sim\\_level](#), the resistor is not taken into consideration for IR drop simulation.

The CustomSim tool automatically adjusts the resistance value based on the simulation strategy. You can override the automatically chosen idealized power-net strategy to meet your accuracy requirements, especially for IR-drop simulations. The smaller the value, the better the accuracy for IR-drop simulations.

The main usage for the `set_powernet_level` command is to use it along with [set\\_sim\\_level](#) to speed up the IR drop simulation. Set [set\\_sim\\_level](#) command based on the circuit types (see [Table 15](#)) and use a higher `set_powernet_level` to override the pre-determined resistance value of [set\\_sim\\_level](#). Otherwise, the resistance value of `set_powernet_level` by default matches the levels by [set\\_sim\\_level](#). For a more detailed IR drop simulation, set a higher `set_powernet_level` to take more parasitic resistors connected to an idealized power-net block for simulation.

*Table 15 Guidelines for Appropriate Circuit Types for Each set\_sim\_level*

Level	Digital	Memory	Low-Sensitivity Analog	High-Sensitivity Analog	Mixed-Signal	Full-Chip
7	X		X	X		X

*Table 15 Guidelines for Appropriate Circuit Types for Each set\_sim\_level*

Level	Digital	Memory	Low-Sensitivity Analog	High-Sensitivity Analog	Mixed-Signal	Full-Chip
6	X	X	X	X	X	X
5	X	X	X	X	X	X
4	X	X	X	X	X	X
3	X	X	X		X	X

### Examples

```
set_sim_level 3
set_powernet_level 6
```

In the previous example, the CustomSim tool overrides only the idealized power-net resistor tolerances used in `set_sim_level` 3 with the tolerances used in `set_sim_level` 6. All resistors connected to power supply with values larger or equal to 1 ohm are taken into the simulation. `set_sim_level` 3 is suitable for IR drop simulation for digital, memory or low-sensitivity analog designs.

## set\_powernet\_option

Controls how resistors are partitioned into the idealized power-net block.

### Syntax

```
set_powernet_option -ideal_rmax res_value
    [-collapse_node node {node}]
    [-except_node node_name {node_name}]
    [-report 1|0]
```

Argument	Description
<code>res_value</code>	Specifies the maximum resistance value of the resistor to be partitioned into the idealized power net block.
<code>-collapse_node node {node}</code>	Collapses the resistor network to the specified node name.

---

Argument	Description
-except_node node_name { node_name }	Specifies a list of nodes whose resistive networks should not be identified as power nets. You can use wildcard characters in the node names.
-report 0   1	Dumps a table of identified power nets in the log file. The default is off.

---

### Description

This command controls how resistors are partitioned into the idealized power-net block. Power-net elements and nodes connected to ground through low-impedance elements (such as voltage sources and resistors smaller than the specified value) are put into the idealized power-net block. [Table 16](#) shows the defaults used at the different accuracy levels.

*Table 16 Default -ideal\_rmax res\_value Values*

---

Accuracy Level	Default -ideal_rmax res_value values
set_sim_level 3	10 ohms
set_sim_level 4	7.5 ohms
set_sim_level 5	5 ohms
set_sim_level 6	1 ohms
set_sim_level 7	1e-5 ohms

---

In some cases the CustomSim tool might automatically adjust the resistance value based on the simulation strategy. To meet your accuracy requirements, especially for IR-drop simulations, you can override the automatically chosen *res\_value* by using the *-ideal\_rmax res\_value* option. The smaller the *res\_value* the better the accuracy for ID-drop simulations.

---

## set\_probe\_option

Controls probing options.

## Syntax

```
set_probe_option -probe_undefined_node enable_value
[-variable_separator separator_char]
[-netlist_probe_control control_value]
[-skip_flat_pl_node control_value]
```

Argument	Description
-probe_undefined_node enable_value	If you enable this option a probe of an undefined signal writes a 0 valued constant signal to the plot file. This option is only recognized in the Eldo netlist format.
-variable_separator separator_char	You use this option to change the separator between the hierarchical path and the Verilog-A variable or parameter name in the plot file to a colon (:). The default separator is the regular hierarchical delimiter (.) or the delimiter set by the <a href="#">set_sim_hierid</a> command.
-netlist_probe_control control_value	Controls whether the CustomSim tool honors the netlist probe statements (v(), i(), and x()).
	<ul style="list-style-type: none"> <li>▪ 0 causes the CustomSim tool to honor netlist probe statements (default).</li> <li>▪ 1 causes the CustomSim tool to ignore all the netlist probe statements and any other commands that trigger probing.</li> <li>▪ 2 causes the CustomSim tool to ignore all the netlist probing statements but to honor all commands that potentially trigger probing (.measure/.print/vec/vcd).</li> </ul>
-skip_flat_pl_node control_value	Controls whether the CustomSim tool ignores the postlayout nodes for wildcard probing. You can specify one of the following values:
	<ul style="list-style-type: none"> <li>▪ 0 (default) to not skip any flattened postlayout nodes for wildcard probing. This is consistent with the current CustomSim tool usage.</li> <li>▪ 1 to skip all flattened postlayout nodes for wildcard probing (name with ':').</li> <li>▪ 2 to skip only internal nodes (ending with :{number}) but keep others, such as instance pins, for wildcard probing.</li> </ul>

## Description

When you specify `-probe_undefined_node` the CustomSim tool creates a signal in the plot file with a constant value of 0V for any explicitly probed node. This feature is only available for Eldo syntax netlists.

## Examples

```
set_probe_option -probe_undefined_node 1
```

Causes probes of undefined signals to be written to the waveform file with a constant value of 0 (zero).

```
set_probe_option -variable_separator :
```

Causes the probes of Verilog-A module variables and parameters to use a colon separator between the hierarchical path and the variable/parameter name. If the following probe is used: `probe_waveform_va -var x1.x2.va_var_name` the signal name in the waveform file is `x1.x2:va_var_name`.

The next example shows how `set_sim_hierid`, `probe_waveform_va` and `-variable_separator` interact.

```
set_sim_hierid -hierid / #sets delimiter to foreshash
probe_waveform_va x1/x2/xva_module/va_var #must use foreshash
delimiter
set_probe_option -variable_separator : # use colon as variable
separator in plot file
#The wdf output is:
# x1/x2/xva_module:va_Va

set_probe_option -netlist_probe_control 1
```

Ignores all probe statements and other statements that may trigger probing.

---

## **set\_probe\_window**

Defines a printing window that reduces the waveform output file size.

### Syntax

```
set_probe_window -window -split value value
{time_start_value time_end} time_start_value [time_end]
```

where:

*time\_end* ::= *time\_end\_value* | end

Argument	Description
<i>time_start_value</i>	Start time.
<i>time_end_value</i>	Stop time.
-split <i>value</i> { <i>value</i> }	Sets the values at which to split output waveform files and save separated waveform files. The default is not to split waveform files as separated outputs.

### Description

This command affects only the output waveform file. It does not have any effect on any measure/extract or any other diagnostic command. If you specify multiple set\_probe\_window commands, the last one takes precedence. Multiple commands do not cause print windows to accumulate.

If the final *time\_end* value is not specified it is assumed to be end. If multiple windows are specified, they cannot overlap. The window end time must be greater than the start time. The window times must be greater than 0.

The CustomSim tool supports the *tstart* option of HSPICE, Eldo, and TISpice, as well as the *outputstart* option of Spectre. *set\_probe\_window* can be used to set the print window.

### Examples

```
set_probe_window 1u
```

The previous example starts writing to the output file at 1us.

#### Example 43

```
.tran 1n 1u 200n
.opt xa_cmd="set_probe_window 500n"
```

Overrides the *.tran* statement because it occurs last. Printing starts at 500ns.

```
.opt xa_cmd="set_probe_window 500n"
.tran 1n 1u 200n
```

The *.tran* statement overrides the *xa\_cmd*. Printing starts at 200ns.

```
set_probe_window 500n
```

Starts printing at 500ns.

```
set_probe_window 100n 300n 400n 1u
```

Prints from 100ns to 300ns and 400ns to 1us.

```
set_probe_window 100n 300n 400n
```

Prints from 100ns to 300ns and 400ns to 1us.

```
set_probe_window -window 10n 20n  
set_probe_window 30n 40n
```

Command precedence is applied and a single print window of 30n to 40n is used.

---

## **set\_ra\_functional\_resistor**

Specifies the model name for the design resistors to be processed by the reliability analysis (RA) flow.

### **Syntax**

```
set_ra_functional_resistor -res_model model_name  
[-layer layer_name]  
[-map_w width_name_in_netlist]
```

Argument	Description
<code>-res_model <i>model_name</i></code>	Specifies either a regular model or macro model. All design resistors that match the <i>model_name</i> are processed for RA.
<code>-layer <i>layer_name</i></code>	Assigns a layer name for the design resistors. The default is to use the <code>-res_model</code> name as the layer name. All design resistors are assumed to have the same layer name.

Argument	Description
<code>-map_w width_name_in_netlist</code>	Assigns the width parameter name and maps it to w. The default is to look for w as the name of the width.

**Description**

Once you specify the model name of the resistors the RA flow automatically creates a virtual SPF net to store the design resistors, generate the current through those resistors, and propagate the current information to Phase II RA analysis.

The following rules apply to the `set_ra_functional_resistor` command.

- You can specify only one model name per command.
- You can use one or more `set_ra_functional_resistor` commands to identify design resistors with a different model name.
- CustomSim-RA analysis requires a back-annotation setup. For EM analysis on design resistors, the same requirement applies.

**Examples**

```
set_ra_option -sigra 1
set_ra_functional_resistor -res_model rhsim_m
load Ba_file -file spf_file
```

Runs a SIGRA simulation that includes a design resistor with the model name `rhsim_m`.

```
set_ra_option -sigra 1
set_ra_functional_resistor -res_model rhsim_m
set_ra_functional_resistor -res_model rhsim_p
load Ba_file -file spf_file
```

Runs a SIGRA simulation that includes design resistors with model names `rhsim_m` and `rhsim_p`.

```
set_ra_option -sigra 1
set_ra_functional_resistor -res_model rhsim_m -layer design_res
load Ba_file -file spf_file
```

Runs SIGRA simulation that includes a design resistor with model name `rhsim_m` and a particular layer name, `design_res`, instead of the default layer name.

```
set_ra_option -sigra 1
set_ra_function_resistor -res_model rhsim_m -map_w wn
set_ra_function_resistor -res_model rhsim_p -map_w wp
load_ba_file -file spf_file
```

Runs a SIGRA simulation that includes a design resistor with the model name rhsim\_m and rhsim\_p. The netlist uses wp and wn to represent width for the corresponding resistors.

---

## **set\_ra\_net**

Specifies the nets to consider for RA.

### **Syntax**

```
set_ra_net [-selectnet net_names]
           [-skipnet net_names]
```

<b>Argument</b>	<b>Description</b>
<code>-selectnet <i>net_names</i></code>	Provides a list of net names to be analyzed in RA. You must specify the full hierarchical BA name, not the schematic one. The default is to consider all back-annotated nets for RA.
<code>-skipnet <i>net_names</i></code>	When selecting RA nets, you can specify both explicit names or use wildcard characters. The net name must be based on the back-annotation net name, not the schematic name.
<code>-skipnet <i>net_names</i></code>	Provides a list of RA net names to be skipped for the RA flow. Wildcard name matching is allowed. If net names are matched by both the <code>-selectnet</code> and <code>-skipnet</code> options, <code>-skipnet</code> has a higher priority. By default no nets are skipped.

### **Examples**

```
set_ra_net -selectnet vdd*
```

Selects only the net that matches vdd.

```
set_ra_net -selectnet * -skipnet vdd gnd
```

Selects all nets, but skips vdd and gnd nets.

## **set\_ra\_net\_type**

Specifies the net types for power net reliability analysis (PWRA) and signal net reliability analysis (SIGRA).

### **Syntax**

```
set_ra_net_type [-pwnet net_names]
[-signet net_names]
```

Argument	Description
-pwnet <i>net_names</i>	Provides a list of power net names to be analyzed by the PWRA flow. You must specify full hierarchical BA net names, not the schematic ones. Wildcard name matching is supported. By default the net type is determined automatically in the CustomSim tool.  When selecting PWRA nets, you can specify both explicit names or use wildcard characters. The net name must be based on the back-annotation net name, not the schematic name.
-signet <i>net_names</i>	Provides a list of signal net names to be analyzed by the PWRA flow. You must specify full hierarchical BA net names, not the schematic ones. Wildcard name matching is supported. By default the net type is determined automatically in the CustomSim tool.  When selecting SIGRA nets, you can specify both explicit names or use wildcard characters. The net name must be based on the back-annotation net name, not the schematic name.

### **Description**

The `set_ra_net_type` command selects the type of net that goes into the RA flow. There are only two types of nets that can go into RA: power nets or

signal nets. If a net is selected as both power and signal, then the power selection takes precedence.

When you do not specify a `set_ra_net_type` command, the CustomSim tool uses internal ID methods to identify net types. When you specify one or more `set_ra_net_type` commands, the automatic identification is disabled, allowing you to control net types manually.

You can specify multiple `set_ra_net_type` commands. Only nets that are selected by `-pwnet` or `-signet` are selected for RA (`set_ra_option -pwra 1` triggers PWRA and `set_ra_option -sigra 1` triggers SIGRA).

### **Examples**

```
set_ra_net_type -pwnet vdd vddhd
```

Defines vdd and vddhd as power nets.

---

## **set\_ra\_option**

Specifies options for RA analysis.

### **Syntax**

```
set_ra_option [-pwra enable_value]
              [-sigra enable_value]
              [-waveform_split size]
              [-rap2auto enable_value]
              [-ratau ratau_value]
              [-ratcl ratcl_name1 ratcl_name2 ...]
              [-twindow time_points]
              [-tmpdir directory_name]
```

---

Argument	Description
<code>-pwra enable_value</code>	Activates the CustomSim-PWRA flow. The default is 0.
<code>-sigra enable_value</code>	Activates the CustomSim-SIGRA flow. The default is 0.

---

Argument	Description
<code>-waveform_split size</code>	Sets the split size for the <code>.rasim</code> file. The default split size is 2G. If the <code>.rasim</code> file size exceeds this threshold, then additional <code>.rasim</code> files are created with names <code>.rasim#</code> , for example, <code>.rasim1</code> , <code>.rasim2</code> , and so on.
<code>-rap2auto enable_value</code>	Launches the phase II RA simulation run automatically. The default value is 1. When you set this option to 0, the phase I simulation stops after the <code>.rasim</code> file is generated.
<code>-ratau ratau_value</code>	Sets the RATAU (simulation step size) value for Phase II RA. The default is 10ps.
<code>-ratcl ratcl_name1 ratcl_name2 ...</code>	Reads the specified <code>ra_tcl</code> file with instructions for phase II RA. You can specify multiple <code>ra_tcl</code> files.
<code>-twindow time_points</code>	Specifies to generate <code>.rasim</code> file data only within time windows defined by:  <code>-twindow tstart tstop {tstart tstop} [tstart]</code>  By default, the CustomSim tool uses <code>tstart=0ns</code> and <code>tstop=end_of_transient</code> as the window.  When you use the <code>-twindow</code> option, if you want a similar window to be used during phase II of reliability analysis, you must use the <code>swin</code> command in the <code>.ratcl</code> file. For more information on the <code>swin</code> command, see the <i>CustomSim Reliability Reference Manual</i> .
<code>-tmpdir directory_name</code>	Specifies the directory name to store temporary files. The default is the current directory.

### Examples

```
set_ra_option -pwra 1 -ratcl ratcl_file
```

Runs the PWRA flow.

```
set_ra_option -pwra 1 -sigra 1 -ratcl ratcl_file
```

Runs both PWRA and SIGRA flows.

```
set_ra_option -pwra 1 -sigra 1 -ratcl ratcl_file -rap2auto 0
```

Runs both PWRA and SIGRA flows and stops after phase I completes.

```
set_ra_option -twindow 5n 10n 20n
```

Collects .rasim data (currents and voltages) in the time interval 5-10ns and from 20ns until the end of transient simulation.

---

## **set\_ra\_pwnet\_driver**

Provides the driver pin information for the internal power net.

### **Syntax**

```
set_ra_pwnet_driver -net net_name
                     -driver_ipname instance_pin_names
```

Argument	Description
<code>-net <i>net_name</i></code>	Provides the back-annotated net name of an internal power net. You can specify only one net name per <code>set_ra_pwnet_driver</code> command. Note: Wildcard characters are not supported.
<code>-driver_ipname <i>instance_pin_names</i></code>	Provides driver instance pins for the specified internal power net. Wildcard characters are not supported.

### **Description**

By default, driver instance pins are found automatically by the CustomSim tool. An output file `.pw_driver` is created that contains information about internal power nets and their drivers. If you specify the `set_ra_pwnet_driver` command, automatic detection is turned off for this net. You can specify multiple `set_ra_pwnet_driver` commands for the same net.

If the net has net pins (\* | P), you should delete those pins if you want only the specified drivers to be used. A solution to delete the unwanted pins is to use the `-delete_netpin` option in the [load\\_ba\\_file](#) command.

**Examples**

```
set_ra_pwnet_driver -net vss -driver_ipname xx/xx/clk2@2:D
```

Defines the driver pin of the vss net.

**set\_ra\_pwnet\_option**

Lets you define the reference voltage and source external power supply node for the internal power net.

**Syntax**

```
set_ra_pwnet_option -net net_name
    [-vref vref] [-src source_pwnet_node]
```

Argument	Description
-net net_name	Provides the back-annotated net name of an internal power net. You can specify only one net name per set_ra_pwnet_option command. Wildcard characters are supported.
-vref vref	Specifies the reference voltage.
-src source_pwnet_node	Specifies the source power net node (its netlist name) that is supplying power into the internal power net. This is a hierarchical node name starting from the top level of the netlist. Wildcard characters are not supported.

**Description**

By default the CustomSim tool automatically identifies internal power nodes and selects appropriate reference voltages and source nodes for them. An output file .pw\_driver is created that contains information about internal power nets and their drivers.

You can override the automatic identification of vref and source\_pwnet\_node with the set\_ra\_pwnet\_option command. If you specify source\_pwnet\_node, the automatic driver ID selects only those drivers that connect to the specified node.

**Examples**

```
set_ra_pwnet_option -net vdd_int -src vdd -vref 0.9
```

Defines the source node and reference voltage of the `vdd_int` net.

---

**set\_ra\_reuse**

Reuses previous simulation results to make the RA verification cycle more efficient.

**Syntax**

```
set_ra_reuse -rasim master_output_path
[-inst_pin_tol tol_value]
```

Argument	Description
<code>-rasim <i>master_output_path</i></code>	Points to the <code>.rasim</code> file location of the master run. The <code>master_output_path</code> is the output directory of the master run. For example, if you use the following command line option for the master run:  <code>xa top.sp -c cmd -o rundir1/csimpl</code> The <code>master_output_path</code> path is set to <code>rundir1/csimpl</code> . It is used as a reference to the master run to get the path to <code>.rasim</code> files. This path must be different from output path in the reuse flow run.

Argument	Description
-inst_pin_tol <i>tol_value</i>	<p>Defines the allowable discrepancy between device pins in the master run and the new DSPF file. Valid value are in the range of 0-1. The default is 0.</p> <p>Consistency in instance pin names is checked between the master run *.rasim files and new DSPF databases. For example, suppose that in PWRA in the master run there are 1K instance pin connections to the VDD net. These instance (or device) pin currents are saved in the *.rasim file in the master run. If the new DSPF has VDD instance pin connections other than those defined in the master run, the simulation checks these connections. The checking is performed at the beginning of phase II simulation. The simulation errors out if the percentage of the discrepancy is greater than user-defined threshold.</p> <p>If the ratio of the number of mismatched device pins and the number of saved device pin currents in the .rasim file exceeds this threshold, the simulation stops. The output mismatch file is generated to the *.pin_mismatch file (same output format as the existing HSIM format).</p>

### Description

Reliability analysis often requires iterations after an initial RA simulation. These iterative results help you debug layout problems. When layout issues are identified and corrected, a new DSPF file is generated and used for the next iteration of simulation. The process of layout changes, generating new DSPF files, and iterative simulation runs might continue until all layout issues are fixed.

Transient analysis can take a significant amount of time in these simulations. To shorten the RA verification cycle, the `set_ra_reuse` command lets you reuse transient analysis results after the first RA run and skip transient analysis for subsequent RA runs.

### Examples

```
set_ra_reuse -rasim output_master/xa
```

Reuses the RASIM result from the master run located in the `output_master` directory.

**Note:** Inside the `./output_master` directory there must be a `xa-pwnet_name.rasim` file available.

---

## **set\_rc\_network\_option**

Controls options related to RC optimization.

### **Syntax**

```
set_rc_network_option [-mode mode_value]
                      [-tconst tconst_value]
```

Argument	Description
<code>-mode mode_value</code>	Controls whether to perform optimization for all parasitic RC networks. You can specify one of the following values. <ul style="list-style-type: none"><li>▪ 0 disables all RC optimizations.</li><li>▪ 1 enables all RC optimizations. This is the default value.</li><li>▪ 2 enables additional RC heuristics for performance throughput.</li></ul>
<code>-tconst tconst_value</code>	Sets the time constant value used by the -mode 2 RC heuristic algorithm. The nodes with a <code>tconst_value</code> greater than the value specified by this option are preserved. The default is 1e-13.

---

## **set\_resistor\_option**

Lets you control resistor handling at different phases of the simulation.

### **Syntax**

```
set_resistor_option -rule 1|2
                     [-min value] [-max value]
                     [-report 0|1]
```

or

```
set_resistor_option [-keep_negative_res enable_value]
[-report 0|1]
```

---

Argument	Description
-rule 1 2	<p>Specifies one of the following optimization rules for resistor elements:</p> <ul style="list-style-type: none"> <li>▪ Specify a value of 1 to process (short or open based on -min and -max values) the resistor elements at the parsing stage, before the optimization and after resolving the parameters. This rule does not save elements in the database, which means you cannot recover them during a simulation. It has the advantage of using less memory.</li> <li>▪ Specify a value of 2 to save the resistor elements in the database and process them during the optimization. The resistors are shorted or opened during optimization. This rule has the advantage of recovering resistor elements for other analysis in the simulation. However, this type of analysis can impact the performance of the simulation.</li> </ul> <p>Note that you can specify both rules in one simulation, but only rule 1 can override rule 1, and rule 2 can override rule 2. Because rule 1 is executed before rule 2, make sure that you specify:  <math>\text{min}(\text{rule1}) &lt; \text{min}(\text{rule2})</math> and <math>\text{max}(\text{rule1}) &gt; \text{max}(\text{rule2})</math>. See <a href="#">Figure 6</a> or further details about specifying both rules in the same simulation.</p> <p>Note that the <code>-rule</code> argument is mandatory and you must specify a value for <code>set_resistor_option</code>.</p>
-min value	Specifies the lower threshold value of the resistors to be kept. All resistors with an absolute value below the specified value are shorted by <code>-rule</code> argument.
-max value	Specifies the upper threshold of the resistors to be kept. All resistors with an absolute value above the specified value are treated as an open-circuit by <code>-rule</code> argument.

Argument	Description
-report 0   1	If set to 1 generates a .rlcignore file with the list of resistor, inductor, and capacitor elements that were optimized by <code>set_resistor_option</code> , <code>set_capacitor_option</code> and <code>set_inductor option</code> . The default is 0 (does not generate a report).
-keep_negative_res enable_value	Enables or disables the control for negative resistors for a simulation. Negative resistor often comes from parasitic extraction. The default is 1 to keep negative resistors (from both a netlist and back-annotation) for optimization by the <code>-rule</code> , <code>-min</code> , and <code>-max</code> arguments. If you want to disable (open) negative resistors for simulation, you need to set this argument.  See <a href="#">Table 17</a> for information about the defaults for handling resistors.

### Description

[Table 17](#) shows the defaults for handling resistors in a simulation.

*Table 17 Defaults for Handling Resistors*

Netlist Format	option=default	<code>set_sim_level=3 through set_sim_level=6</code>	<code>set_sim_level=7</code>
HSPICE	RESMIN=1e-5	Specifies the following defaults: <ul style="list-style-type: none"> <li>▪ If the resistance value is between -1e-5 and 1e-5, those resistors are shorted by <code>set_resistor_option -rule 2</code>.</li> <li>▪ if the resistance value is larger than or equal to 1e12, or is smaller than -1e12, those resistors are open by <code>set_resistor_option -rule 2</code>.</li> </ul>	Specifies the following defaults: <ul style="list-style-type: none"> <li>▪ Set to -1e-5 for negative resistors and 1e-5 for positive resistors.</li> <li>▪ Short resistors with a resistance value of 0.</li> <li>▪ If the resistance value is larger than or equal to 1e12, or is smaller than -1e12, those resistors are open by <code>set_resistor_option -rule 2</code>.</li> </ul>

*Table 17 Defaults for Handling Resistors*

<b>Netlist Format</b>	<b>option=default</b>	<b>set_sim_level=3 through set_sim_level=6</b>	<b>set_sim_level=7</b>
Eldo	RSMALL=1e-6	<p>Specifies the following defaults:</p> <ul style="list-style-type: none"> <li>▪ If the resistance value is between -1e-6 and 1e-6, those resistors are shorted by <code>set_resistor_option -rule 2</code>.</li> <li>▪ if the resistance value is larger than or equal to 1e12, or is smaller than -1e12, those resistors are open by <code>set_resistor_option -rule 2</code>.</li> </ul>	<p>Specifies the following defaults:</p> <ul style="list-style-type: none"> <li>▪ Set to -1e-6 for negative resistors and 1e-6 for positive resistors.</li> <li>▪ Short resistors with a resistance value of 0.</li> <li>▪ If the resistance value is larger than or equal to 1e12, or is smaller than -1e12, those resistors are open by <code>set_resistor_option -rule 2</code>.</li> </ul>
Spectre	rmin=0	<p>Specifies the following defaults:</p> <ul style="list-style-type: none"> <li>▪ If the resistance value is 0, those resistors are shorted by <code>set_resistor_option -rule 2</code>.</li> <li>▪ if the resistance value is larger than or equal to 1e12, or is smaller than -1e12, those resistors are open by <code>set_resistor_option -rule 2</code>.</li> </ul>	<p>Specifies the following defaults:</p> <ul style="list-style-type: none"> <li>▪ Short resistors with a resistance value of 0.</li> <li>▪ If the resistance value is larger than or equal to 1e12, or is smaller than -1e12, those resistors are open by <code>set_resistor_option -rule 2</code>.</li> </ul>

Table 17 Defaults for Handling Resistors

Netlist Format	option=default	<code>set_sim_level=3</code> <code>through</code> <code>set_sim_level=6</code>	<code>set_sim_level=7</code>
Back-annotation (DSPEF or SPEF with <code>load_ba_file</code> )	<code>min_res=0.01</code> <code>max_res=1e12</code>	<p>Specifies the following defaults:</p> <ul style="list-style-type: none"> <li>▪ If the minimum back-annotation resistance value is between -0.01 and 0.01, those resistors are shorted by <code>set_ba_option</code>.</li> <li>▪ if the maximum back-annotation resistance value is larger than or equal to 1e12, or is smaller than -1e12, those resistors are open by <code>set_ba_option</code>.</li> </ul>	Same as <code>set_sim_level=3</code> through <code>set_sim_level=6</code> .

## Examples

```
set_resistor_option -rule 1 -min 0.1 -max 1e10
```

Sets the minimum and maximum threshold to 0.1 and 1e10 for all schematic resistors. By default, all the schematic resistor values between -0.1 and 0.1 are shorted by `-rule 1`. All the schematic resistors with values less than or equal to -1e10, or greater than or equal to 1e10 are open by `-rule 1`.

```
set_resistor_option -rule 1 -min 0.1 -max 1e10
set_ba_option -min_res 0.01
```

Sets the minimum and maximum threshold to 0.1 and 1e10 for all schematic resistors. The minimum threshold for back-annotation resistors is set to 0.01 by `set_ba_option`. By default, all the schematic resistor values between -0.1 and 0.1 are shorted by `-rule 1`. All the schematic resistors with values less than or equal to -1e10, or greater than or equal to 1e10 are open by `-rule 1`.

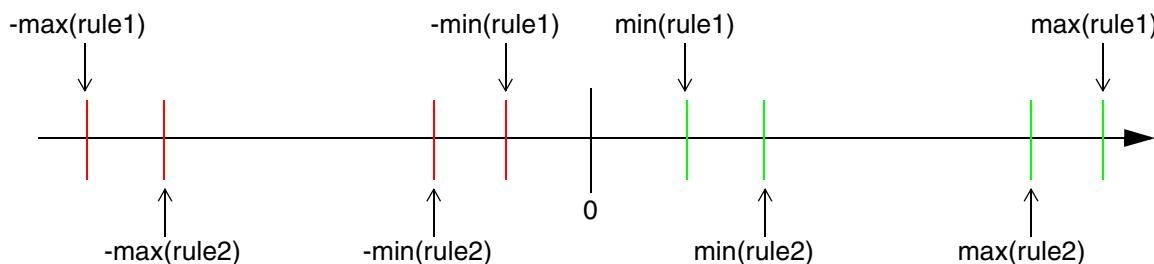
```
set_resistor_option -rule 1 -min 0.1 -max 1e10
set_ba_option -min_res 0.01
load_ba_file -file ba1.spf
load_ba_file -file ba2.spf -min_res 1 -min_cap 1e-20
```

Sets the minimum and maximum threshold to 0.1 and 1e10 for all schematic resistors. The minimum threshold for back-annotation resistors is set to 0.01 by [set\\_ba\\_option](#). The [load\\_ba\\_file -res\\_min](#) command overrides the minimum threshold for the ba2.spf file to 1. By default, all the schematic resistor values between -0.1 and 0.1 are shorted by -rule 1. All the schematic resistors with values less than or equal to -1e10, or greater than or equal to 1e10 are open by -rule 1.

```
set_resistor_option -rule 1 -min 0.5
set_resistor_option -rule 2 -min 10
load_ba_file -file ba.spf
```

All the schematic resistor values between -0.5 and 0.5 are shorted according to -rule 1. All the schematic resistors with values less than or equal to -1e12, or greater than or equal to 1e12 are open according to -rule 1. All the schematic resistor values between 0.5 and 10 and -10 and -0.5 are shorted according to -rule 2.

[Figure 6](#) shows the relationships between -rule 1 and -rule 2 in the same simulation.



(min of rule1) < (min of rule2) and (max of rule1) > (max of rule2) for positive resistors  
 -(min of rule1) > -(min of rule2) and -(max of rule1) < -(max of rule2) for negative resistors

*Figure 6 Using -rule 1 and -rule 2 for resistors in the same simulation*

### See Also

[load\\_ba\\_file](#)  
[set\\_ba\\_option](#)

## **set\_restore\_option**

Restarts the saved run at time 0, regardless of the saved time.

### **Syntax**

```
set_restore_option -time 0
```

---

Argument	Description
-time 0	Note that the CustomSim tool does not do any time shift of the input stimuli, it just restarts the run at time 0.

---

## **set\_sample\_point**

Use this command when you need to make very precise periodic measurements, such as in FFT applications.

### **Syntax**

```
set_sample_point -period period_value
    [-twindow {start_time stop_time} start_time [stop_time]]
```

---

Argument	Description
-period <i>period_value</i>	Sets the period between sampling points (the sampling frequency). If you specify multiple <i>start_time</i> and <i>stop_time</i> arguments, the period applies to all the -twindow values.
-twindow { <i>start_time</i> <i>stop_time</i> } <i>start_time</i> [ <i>stop_time</i> ]	Sets the start and stop time of the first sampling point.

---

### **Description**

Very precise measurements such as those made during FFT analysis can be adversely affected by sampling the simulator output waveforms, if the sample points interpolate between time points solved by the simulator. This command forces the simulator to synchronize all partitions and solve each time point that is sampled during a post-processing measurement. This prevents any

interpolation errors and maximizes the precision of a measurement.

### Examples

Assume you are simulating a DAC and need to analyze the spectral output by computing an FFT. The following example starts the first sample at 10 microseconds and sets the sampling rate to 10 mega samples per second. That is, a sample point occurs at 10 $\mu$ , 10.1 $\mu$ , 10.2 $\mu$ , 10.3 $\mu$ , and so forth.

```
set_sample_point -period 100n -twindow 10u
```

Assume you are simulating a DAC and need to analyze the spectral output by computing an FFT. In the following example, the first sample is at 10 $\mu$  sampling at a rate of 100Msps. The end of the sampling time is 20 $\mu$ .

```
set_sample_point -period 100n -twindow 10u 20u
```

## **set\_save\_state**

Saves a partial simulation run to be restarted later.

### Syntax

```
set_save_state [-time t_val {t_val}] |  
[-period time_period] |  
[-period_wall_time wall_time]  
[-save_on_kill enable_value]  
{ [-time t_val {t_val}] |  
[-period time_period] |  
[-period_wall_time wall_time]  
[-save_on_kill enable_value] }  
[-type op|OP] [-file file_name]  
[-dump_waveform 0|1]
```

Argument	Description
-time <i>t_val</i> { <i>t_val</i> } ...	Saves the simulation at the time or times you specify. For each <i>t_val</i> time you specify the CustomSim tool creates a file with <i>.t_val#.ic</i> extension. The <i>dc</i> (for time 0) and <i>end</i> (for last transient point) keywords are valid values for <i>t_val</i> .
-period <i>time_period</i>	Saves the simulation at the time period you specify. The image is saved in a file with a <i>.time.ic</i> extension.

Argument	Description
<code>-period_wall_time wall_time</code>	Saves the simulation in a single file at the "wall time" period you specify. The image is saved with a <code>.save.ic</code> extension. The file is overwritten at each specified period time interval. You must specify the wall time period in hours, and decimal points are accepted.
<code>-save_on_kill enable_value</code>	Saves the simulation when run a UNIX <code>kill -15</code> command. You cannot use a <code>kill -9</code> command. The image is saved in a file with a <code>.save.ic</code> extension.
<code>-type op OP</code>	Specifies the saving type. Only <code>op</code> is currently available.
<code>-file file_name</code>	Specifies the names of the saved files. The saved files have <code>.ic</code> and <code>.ic.sup</code> extensions. If you specify multiple filenames, the last one is used for all saved files.
<code>-dump_waveform 0 1</code>	Saves fsdb files at each periodic save time specified by <code>-period</code> . The default is 0.

**Description**

The `set_save_state` command generates two files, one with a `.time.ic` extension and one with a `.time.ic.sup` extension for each saved time you specify with the `-time` or `-period` option.

When you specify `-period_wall_time`, two files are created with a `.save.ic` and `.save.ic.sup` extension. The files are overwritten at each wall time period. When you specify `-save_on_kill`, two files are created with a `.save.ic` and `.save.ic.sup` extension.

**Note:** Do not delete the `.ic` or `.ic.sup` files. You need both files to restore a saved simulation. For more information about restoring a saved simulation, see the *CustomSim User Guide*.

The following rules apply to diagnostics commands and measurements:

- The diagnostic files (.errz, .errt, .hotspot, and CCK command outputs) must be written out and updated at each saved time point.
- Output measurements (.power, .meas, .mt) that have completed must be written out and updated at each saved time point.
- When restoring a simulation, if a measurement command starts before the restore time and finishes after the restore time, that measurement is ignored and a warning is issued.

The `set_save_state` command has the following restrictions:

- You cannot specify multiple `-time`, `-period` and `-period_wall_time` options on a single command line.
- You can specify multiple values for `-time`. Only one value is allowed for the other options.
- You cannot specify multiple `set_save_state` commands with different `-period` or `-period_wall_time` values. If multiple commands with multiple `-period` or `-period_wall_time` are specified, the last period or `wall_time` is applied.
- You can specify multiple `set_save_state` commands with different `-time` options.
- You can specify one `set_save_state` command with `-period` or `-period_wall_time` and one or more `set_save_state` commands with the `-time` option.
- You can combine the `-save_on_kill` option with the other options.

### Examples

```
set_save_state -time 100n 1400n -type OP
```

Saves the simulation at 100ns and at 1400ns.

```
set_save_state -period 200n -type OP
```

Saves the simulation every 200ns.

```
set_save_state -period_wall_time 0.1 -type OP
```

Saves the simulation every 0.1 hours (every 6 minutes).

```
set_save_state -save_on_kill 1 -type OP
```

Saves the simulation when the run is killed with the UNIX `kill -15` command.

```
set_save_state -period 200n -type OP  
set_save_state -time 10n 20n -type OP
```

Saves the simulation at 10ns, 20ns, and every 200ns.

```
set_save_state -time 300n -type OP  
set_save_state -time 10n 20n -type OP
```

Saves the simulation at times 10ns, 20ns and 300ns.

---

## **set\_sim\_case**

Controls case-sensitivity. Note that you must specify this command in the CustomSim configuration file, not in the netlist.

### **Syntax**

```
set_sim_case -case upper|lower|sensitive
```

Argument	Description
<code>-case</code> <code>upper lower sensitive</code>	Specify one of the following keywords: <ul style="list-style-type: none"><li>■ <code>upper</code> to convert all the names to uppercase.</li><li>■ <code>lower</code> to convert all the names to lowercase.</li><li>■ <code>sensitive</code> to treat all names as case-sensitive.</li></ul> The default is based on the netlist format. For HSPICE and Eldo, the default is lowercase. For TISPACE, the default is uppercase. For Spectre and Verilog, the default is case-sensitive.

### **Description**

`set_sim_case` lets you control the case-sensitivity of the CustomSim tool for all the supported netlist formats. When the `-case upper|lower` argument is set, the CustomSim tool is case-insensitive and converts all names to uppercase or lowercase, respectively. The `-case sensitive` argument enables the CustomSim tool to be case-sensitive for all netlist formats.

**Note:** The `temper` and `hertz` keywords are always case insensitive when evaluating expressions.

### Examples

```
* Example 2 : HSPICE format
R1 port1 0 10
R2 porT1 0 10
r3 Port1 0 10

.probe v(*) i(r*)
```

If you specify `set_sim_case -case upper` or `-case lower`, the CustomSim tool is case-insensitive and all 3 resistors are connected to the same node, `port1`. If you specify `set_sim_case -case sensitive`, case-sensitivity is enabled and the 3 resistors are connected to 3 different nodes: `port1`, `porT1`, and `Port1`.

## set\_sim\_hierid

Specifies the hierarchical separation character.

### Syntax

```
set_sim_hierid -hierid sep_char [-add_spf sep_char]
```

Argument	Description
<code>-hierid <i>sep_char</i></code>	The legal hierarchical separation characters are: <ul style="list-style-type: none"> <li>▪ Period (.) (default)</li> <li>▪ Colon (:)</li> <li>▪ Forward slash (/)</li> <li>▪ Caret (^)</li> </ul>
<code>-add_spf <i>sep_char</i></code>	This <code>-add_spf</code> option effectively replaces the SPF netlist delimiter with the <code>set_sim_hierid</code> delimiter. The replacement delimiter is applied in all included files. This option can be useful when you include a back-annotation netlist, as in a flat flow.

**Description**

If you specify both the Eldo .hier option and the set\_sim\_hierid command in the netlist file, the set\_sim\_hierid command takes precedence. The same precedence applies if the set\_sim\_hierid command is in the CustomSim command file.

**Examples**

```
set_sim_hierid /
```

Sets the hierarchical separation character to "/".

```
set_sim_hierid :
```

Sets the hierarchical separation character to ":".

---

**set\_sim\_level**

Controls the speed and model complexity trade-off.

**Syntax**

```
set_sim_level -level level instance_spec
```

---

Argument-- level	Description
3	Specifies a functional and timing verification of digital, memory, low-sensitivity analog, mixed-signal, and full-chip circuits.
4	Specifies a functional, timing and power verification of all circuits, especially for small current or low voltage applications.
5	Specifies an accurate timing and power simulation of all circuits, and block characterization.
6	Specifies a SPICE-like accuracy for timing and power simulation of all circuits, and cell characterization.
7	Specifies a setting for small designs (less than 1000 elements) or a small block of a large design, and device model verification.

---

## Description

See the [Common Syntax Definitions](#) section for details about the *instance\_spec* argument.

Controls the simulator speed and model complexity tradeoff. This command can be applied to the entire netlist, or to specific subcircuits or instances. The **default** level is 3, if this command is not specified.

**Note:** The local specification of the `set_sim_level` command always takes precedence over the global setting.

## Examples

[Example 44](#) sets the simulation level to 5 on the entire netlist:

*Example 44*

```
set_sim_level 5
```

[Example 45](#) sets the simulation level on a specific instance:

*Example 45*

```
set_sim_level 4 -inst xpll
```

[Example 46](#) sets the simulation level on all instances of a particular subcircuit type:

*Example 46*

```
set_sim_level 6 -subckt chgpump
```

---

## set\_sram\_characterization

Specifies the performance/accuracy settings for simulating SRAM designs.

### Syntax

```
set_sram_characterization [-enable] enable_value
  [-ver[ision] 0|1]
  [-app[lication] timing|power]
  [-acc[uracy] 1|2|3|4|5]
  [-resistor_rule 1|2]
  [-capacitor_rule 1|2]
```

Argument	Description
<code>enable_value</code>	Turns on the <code>set_sram_characterization</code> command.
<code>-version 0 1</code>	<p>Specify the <code>-version</code> argument according to the process node of the design:</p> <ul style="list-style-type: none"> <li>▪ 0 is the recommended setting for designs that use the process nodes of 14nm and above. This is the default setting.</li> <li>▪ 1 is the recommended setting for designs that use the process nodes of 10nm and below. Note that setting <code>-version</code> to 1 enables multi-core simulation with a default of two cores. You can adjust the number of cores used with the <code>-mt</code> command line option, or by adding <code>set_multi_core -core</code> to the configuration command file.</li> </ul>
<code>-application timing power</code>	<p>Specifies the SRAM simulation types:</p> <ul style="list-style-type: none"> <li>▪ <code>timing</code> for a timing simulation (default).</li> <li>▪ <code>power</code> for a power or leakage simulation.</li> </ul> <p>Note that use can use the <code>app</code> alias for this option. The name is case insensitive.</p>
<code>-accuracy 1 2 3 4 5</code>	<p>Specifies the performance/accuracy tradeoff. The default of 3 provides the best combination of performance/accuracy. Specify 1 for the fastest simulation, or 5 for the most accurate.</p> <p>Note that use can use the <code>acc</code> alias for this option.</p>
<code>-resistor_rule 1 2</code>	<p>You can specify one of the following values to control how the CustomSim tool processes resistors:</p> <ul style="list-style-type: none"> <li>▪ 1 (default) to specify that resistor optimizations are done during parsing.</li> <li>▪ 2 to specify that resistor optimizations are done during front-end optimization. Specify this option when you want to protect the resistors that are probed.</li> </ul>

Argument	Description
-capacitor_rule 1   2	You can specify one of the following values to control how the CustomSim tool processes capacitors: <ul style="list-style-type: none"> <li>▪ 1 (default) to specify that capacitor optimizations are done during parsing.</li> <li>▪ 2 to specify that capacitor optimizations are done during front-end optimization. Specify this option when you want to protect the capacitors that are probed.</li> </ul>

**Description**

For details about how to use the `set_sram_characterization` command to simulate SRAM designs, see the *CustomSim User Guide*.

**Important:** Do not use `set_sram_characterization` with the `set_sim_level` command in the same configuration file. This combination causes the CustomSim tool to issue a warning and ignore the `set_sim_level` command.

**Examples**

```
set_sram_characterization 1 -application timing
```

Specifies an SRAM timing simulation with the default setting.

```
set_sram_characterization 1 -application timing -accuracy 4
```

Specifies an SRAM timing simulation with accuracy level 4.

```
set_sram_characterization 1 -app timing -acc 4
```

Same as the previous example, but uses an abbreviated name.

```
set_sram_characterization 1 -application power
```

Specifies an SRAM power simulation with the default setting.

```
set_sram_characterization 1
```

Specifies an SRAM simulation with the default settings.

## set\_synchronization\_level

Controls the synchronization settings between the gate/channel/bulk of the elements.

### Syntax

```
set_synchronization_level -level level_value  
[-inst inst_name {inst_name}]  
[-subckt subckt_name {subckt_name} ]
```

---

Argument	Description
-level level_value	Sets the synchronization level. By default, the set_synchronization_level value matches the specified <a href="#">set_sim_level</a> value.
-inst inst_name {inst_name}	Applies the synchronization settings to the specified instances.
-subckt subckt_name {subckt_name}	Applies the synchronization settings to the specified subcircuits.

### Description

You use the `set_synchronization_level` command with [set\\_sim\\_level](#) to speed up or to increase the accuracy of a simulation. For example, in some designs that use `set_sim_level 5`, you can get better performance, and close to the same accuracy, with `set_synchronization_level 3`.

---

## set\_synchronization\_option

Provides flexibility to control the synchronization between blocks.

### Syntax

```
set_synchronization_option -rule gate|never  
[-inst inst_name {inst_name}]  
[-subckt subckt_name {subckt_name} ]
```

Argument	Description
-rule gate never	gate specifies that If the gate of a MOSFET is in a different partition than the channel, those two partitions are evaluated at the same time points. Applying synchronization on the gate of this MOSFET might help to improve accuracy. never turns off the synchronization between the different partitions. This is useful for speeding up the simulation when the signal is only fed forward from gate to the channel when <code>set_partition_option -rule</code> is applied.
-inst <i>inst_name</i> { <i>inst_name</i> }	Applies the synchronization rule to the specified instances.
-subckt <i>subckt_name</i> { <i>subckt_name</i> }	Applies the synchronization rule to the specified subcircuits.

**Description**

This command is often used with the [set\\_partition\\_option](#) command. When manual partitioning is applied to an instance, different partitions might need to be synchronized to improve the accuracy, or un-synchronized to speed up the simulation. You can apply this command can be applied globally or locally.

---

**set\_tolerance\_level**

Provides flexibility for tuning the CustomSim tool relative tolerance level to the [set\\_sim\\_level](#) setting. The relative tolerance level controls the sensitivity of the simulator to small voltage changes.

**Syntax**

```
set_tolerance_level -level tolerance_level [instance_spec]
```

Argument	Description
<code>-level</code> <code>tolerance_level</code>	<p>The default relative tolerance levels that correspond to the <a href="#">set_sim_level</a> settings are as follows:</p> <ul style="list-style-type: none"> <li>▪ 3 applies the same relative tolerance value used with <a href="#">set_sim_level</a> 3.</li> <li>▪ 4 applies the same relative tolerance value used with <a href="#">set_sim_level</a> 4.</li> <li>▪ 5 applies the same relative tolerance value used with <a href="#">set_sim_level</a> 5.</li> <li>▪ 6 applies the same relative tolerance value used with <a href="#">set_sim_level</a> 6.</li> <li>▪ 7 applies the same relative tolerance value used with <a href="#">set_sim_level</a> 7.</li> </ul> <p>The following section describes the cases when a lower or higher <code>tolerance_value</code> can improve performance and accuracy.</p>
<code>instance_spec</code>	See the <a href="#">Common Syntax Definitions</a> section for details about the <code>instance_spec</code> argument.

### Description

`set_tolerance_level` provides more flexibility to tune the CustomSim tool to specific simulation requirements, similar to [set\\_tolerance\\_option](#). It lets you adjust the relative tolerance level appropriate to the corresponding [set\\_sim\\_level](#) level.

The `set_tolerance_level` command is helpful when:

- A CustomSim simulation is functionally correct at a lower [set\\_sim\\_level](#) setting, but a higher [set\\_sim\\_level](#) setting is needed to achieve the required accuracy. There are cases when you can use a lower [set\\_sim\\_level](#) setting with a higher `tolerance_level` to obtain the desired accuracy with better performance.  
For example, if a data converter is functional at `set_sim_level=5` and higher, but the FFT requirements are only met at `set_sim_level=7`. You might find that `set_sim_level=5` and `set_tolerance_level 7` achieves the required accuracy with significantly better performance than `set_sim_level=7`.
- An CustomSim simulation meets the accuracy requirements, but runs slowly. You can set `tolerance_level` to a lower value below the corresponding `set_sim_level` setting to gain better performance with

some sacrifice in accuracy. A lower `tolerance_level` setting might be useful when another simulator gets good results with or close to its default settings, but the CustomSim tool runs more slowly.

`set_tolerance_level` can control the relative tolerance level globally or locally. The most significant impact of the relative tolerance level is to adjust the time step of the simulator. There are secondary impacts on MOS lookup table granularity and RC reduction.

### Examples

```
set_sim_level 6
set_tolerance_level 7 -subckt a2d
```

This example applies a `set_sim_level` of 6 setting to the entire circuit. For the `a2d` subcircuit, the tolerance parameters are overwritten with a more conservative setting of `set_tolerance_level` 7.

## set\_tolerance\_option

Lets you control the simulation tolerance independent of the `set_sim_level` command for more flexibility in tuning performance/accuracy.

### Syntax

```
set_tolerance_option -tol tol_value
  [-tol_rule 0|1]
  [-inst inst_name {inst_name}]
  [-subckt subckt_name {subckt_name}]
```

Argument	Description
<code>-tol tol_value</code>	Specifies the tolerance value. The most significant impact of this value is to adjust the time step of the simulation. However, the specified tolerance value also has secondary impacts on the MOS look-up table granularity and RC reduction. <a href="#">Table 18</a> shows the default tolerance values.

Argument	Description
<code>-tol_rule 0 1</code>	Applies the CustomSim simulator heuristics as follows: <ul style="list-style-type: none"> <li>▪ A value of 0 applies the specified <code>tol_value</code> as a base setting, but the simulator heuristics might override this value in certain parts of the design. The value override used by the simulator heuristics is typically smaller than the user-specified <code>tol_value</code>.</li> <li>▪ A value of 1 applies the specified <code>tol_value</code> to the entire design. The simulator heuristics do not override the <code>tol_value</code>.</li> </ul>
<code>-inst inst_name {inst_name}</code>	Instance names to which <code>set_tolerance_option</code> applies.
<code>-subckt subckt_name {subckt_name}</code>	Subcircuit names to which <code>set_tolerance_option</code> applies.

### Description

By providing more flexibility in tuning a CustomSim simulation to specific design requirements, `set_tolerance_option` lets you control the simulation tolerance independent of the [set\\_sim\\_level](#) command. You can apply `set_tolerance_option` globally or locally.

The tolerance value (`-tol`) controls the sensitivity of the simulation to small voltage changes. This value is arbitrary and is not based on a unit. Valid values are 0.1 to 800. Note that a value of 800 is extremely aggressive in favor of performance, and most circuits are not functional with this setting.

The default tolerance values for each `set_sim_level` setting are shown in [Table 18](#).

*Table 18 Default Relative Tolerance Values*

Accuracy Level	Default <code>set_tolerance_option</code> Value
<code>set_sim_level 3</code>	<code>-tol=200</code> (less accurate)
<code>set_sim_level 4</code>	<code>-tol=150</code>
<code>set_sim_level 5</code>	<code>-tol=100</code>
<code>set_sim_level 6</code>	<code>-tol=50</code>

**Table 18 Default Relative Tolerance Values**

<b>Accuracy Level</b>	<b>Default set_tolerance_option Value</b>
set_sim_level 7	-tol=0.1 (very accurate)

The `set_tolerance_option` command is helpful when:

- An CustomSim simulation is functionally correct at a lower `set_sim_level` setting, but a higher `set_sim_level` setting is needed to achieve the required accuracy. There are cases when you can use a lower `set_sim_level` setting with a smaller `tol_value` to obtain the desired performance and accuracy. For example, if a data converter is functional at `set_sim_level=5` and higher, but the FFT requirements are only met at `set_sim_level=7`, you might find that `set_sim_level=5` and `set_tolerance_option -tol 0.1` achieve the required accuracy with performance significantly better than `set_sim_level=7`.
- An CustomSim simulation meets the accuracy requirements, but runs slowly. You can increase `tol_value` above the default `set_sim_level` setting to gain better performance with some sacrifice in accuracy. A higher `tol_value` setting might be useful when another simulator gets good results with or close to its default settings, but CustomSim runs more slowly.

### Examples

```
set_sim_level 6
set_tolerance_option -tol 0.1 -subckt a2d
```

This example applies `set_sim_level=6` to the entire circuit, but overrides the tolerance parameters for the `a2d` subcircuit with a more conservative setting of 0.1.

## **set\_va\_view**

Controls whether a SPICE subcircuit or a Verilog-A module definition (or “view”) is used for the entire netlist, or for an instance or subcircuit when both SPICE and Verilog-A definitions exist for that instance or subcircuit.

## Syntax

```
set_va_view -view VA|va|SPICE|spice
[-inst inst_name {inst_name}]
[-subckt subckt_name {subckt_name}]
```

Argument	Description
-view VA   va	Specifies that Verilog-A module definitions have priority over SPICE subcircuit definitions.
SPICE   spice	Specifies that SPICE subcircuit definitions have priority over Verilog-A module definitions.
-inst inst_name {inst_name}	Specifies a particular instance or set of instances the set_va_view command affects. See <a href="#">Common Syntax Definitions</a> for more about specifying instance names.
-subckt subckt_name {subckt_name}	Specifies a particular subcircuit or set of subcircuits the set_va_view command affects. See <a href="#">Common Syntax Definitions</a> for more about specifying subcircuit names.

## Description

If your design has both SPICE subcircuit definitions and Verilog-A definitions, you can switch between SPICE and Verilog-A depending on the type of analysis you want to perform. This lets you achieve tradeoffs between the speed and accuracy of the simulation for particular definitions. Use the set\_va\_view command to perform module- and instance-based partitioning of the circuit definitions. You can switch to a Verilog-A module description for a single SPICE subcircuit, or set of subcircuits, within a netlist.

If multiple definitions or “views” of a SPICE subcircuit or Verilog-A module exist, the CustomSim tool defaults to use the view of the parent netlist. If you set the entire netlist to use SPICE (set\_va\_view -view SPICE), then any subcircuit instance uses the subcircuit definition by default. If you set the entire netlist to use Verilog-A (set\_va\_view -view VA), then the Verilog-A module definitions is used by default; if the module definition does not exist then the subcircuit view is used.

If you specify set\_va\_view for a specific subcircuit or instance, then that subcircuit or instance uses the definition you specify (SPICE or Verilog-A), as shown in the following examples.

If set\_va\_view references a view that does not exist, a warning message appears and the other view is used. If neither a SPICE nor a Verilog-A

definition exist for the subcircuit, the CustomSim tool returns an error and the simulation is terminated.

**Note:** If an Eldo instance is declared with a `y` element, the Verilog-A definition is used in all cases. It cannot be overridden with the `set_va_view` command.

## Examples

[Example 47](#) shows an HSPICE netlist that uses the SPICE view by default.

*Example 47 Netlist using SPICE definitions by default*

```
.hdl mymodule.va
.subckt mymodule a b
...
.ends mymodule
X1 1 2 mymodule
X2 3 4 mymodule
X3 5 6 mymodule
```

The command in [Example 48](#) sets the SPICE view for all subcircuits and modules in a netlist.

*Example 48*

```
set_va_view SPICE
```

The command in [Example 49](#) sets the Verilog-A view for all subcircuits and modules in a netlist.

*Example 49*

```
set_va_view VA
```

In [Example 50](#), the Verilog-A module definition is used for all instances of subcircuit *my\_module*.

*Example 50*

```
set_va_view VA -subckt mymodule
```

If the netlist contains the command in [Example 51](#), the Verilog-A module definition is used for subcircuit instance X2 of *my\_module*.

*Example 51*

```
set_va_view VA -inst X2
```

**See Also**

[“Netlist Syntax for Verilog-A in CustomSim”](#) in the *CustomSim User Guide* for details about how to include Verilog-A definitions in HSPICE, Spectre, or Eldo netlists.

---

## **set\_vector\_char**

Provides a capability to overwrite the logic-high and logic-low voltages specified in the vector file without the need to modify the original vector file. The arguments supported in this command are used globally.

**Syntax**

```
set_vector_char [-vih value]
                 [-vil value]
                 [-node node_name {node_name}]
```

Argument	Description
-vih value	Overwrites the logic-high voltage for input vectors or bidirectional vectors when in input mode. The default value is set to 3.3 volts.
-vil value	Overwrites the logic-low voltage for input vectors or bidirectional vectors when in input mode. The default value is set to 0.0 volts.
-node node_name {node_name}	Specifies a node name inside a vector file. You can specify multiple node names with wildcard characters. The default is to match all nodes specified in the vector file.

**Examples**

```
set_vector_char -vih 3.0 -vil 0.8
```

Overwrites logic-high and logic-low voltage defined in the vector file by 3.0 and 0.8 respectively.

## set\_vector\_option

Allows the plotting of a logic signal to represent the "output expected" signal for any specified output variable inside the VEC/VCD file.

**Note:** Only the WDF, FSDB, OUT and VPD formats support this feature.

### Syntax

```
set_vector_option [-check_u_state value]
                  [-check_z_state value]
                  [-check_method value]
                  [-define_x_state state_value]
                  [-vec_mode mode]
                  [-precedence error|vsrc ]
                  [-print_expected switch_value
                   [-node node_name {node_name}]]
                  [-check_io message]
```

Argument	Description
-check_u_state value	<p>Checks the U-state between the signals in the VCD and CustomSim VEC files against the actual state of the nodes. You can specify a value of 0 or 1:</p> <ul style="list-style-type: none"> <li>▪ 0 (default) specifies that if a node is in a U-state, it passes the vector checking regardless of the value in the VCD or CustomSim VEC files. In addition, if U-state is specified in the VCD or CustomSim VEC files, a node passes checking regardless of its state.</li> <li>▪ 1 specifies that the node state level must match the expected U-state specified in the CustomSim VEC file. If a VCD file is used, the check_u_state check is ignored and no checking is performed.</li> </ul>

Argument	Description
<code>-check_z_state value</code>	Checks the Z-state between the signals in the VCD and CustomSim VEC files against the actual state of the nodes. You can specify a value of 0 or 1: <ul style="list-style-type: none"> <li>▪ 0 (default) specifies that the floating status for nodes is not checked.</li> <li>▪ 1 specifies that floating status for nodes must match the floating status specified in VCD and CustomSim VEC files.</li> </ul>
<code>-check_method value</code>	Specifies the vector checking mechanism between the VCD and VEC formats. You can specify a value of 1 or 2: <ul style="list-style-type: none"> <li>▪ 1 specifies continuous checking, which is the default for the VCD format. This check method only supports the VCD format.</li> <li>▪ 2 specifies strobe-based time point checking, which is the default method for the HSPICE VEC format. This check method supports both HSPICE VEC and VCD formats.</li> </ul>
<code>-define_x_state state_value</code>	Defines how to treat the X state. The acceptable <i>state_value</i> is range from 1 through 4: <ul style="list-style-type: none"> <li>▪ 1 to always set to low_voltage.</li> <li>▪ 2 to always set to high_voltage.</li> <li>▪ 3 to set (high_voltage + low_voltage)/2.</li> <li>▪ 4 to set the voltage value to the previous voltage.</li> </ul> The default <i>state_value</i> is set to 1. When the x-state is set, it also reflects the printing of the logic waveform on the x-state as well.
<code>-vec_mode mode</code>	Specifies one of the following formats: <ul style="list-style-type: none"> <li>▪ hspice to use the HSPICE vector file format (default).</li> <li>▪ hsim to use the HSIM file vector file format.</li> </ul>

Argument	Description
<code>-precedence error vsrC</code>	Determines whether a voltage stimulus defined in the netlist file can overwrite a vector input (or biput) from a VEC file if both are applied to the same nodes. You can specify one of the following values: <ul style="list-style-type: none"> <li>▪ <code>error</code> (default) causes the CustomSim tool to error out if it detects duplicated voltage sources from the netlist and VEC files.</li> <li>▪ <code>vsrC</code> causes the CustomSim tool to use the voltage source from the stimulus file (for example, netlist) if duplicated voltage sources are specified in both stimulus file and VEC vector file on the same node,</li> </ul>
<code>-print_expected switch_value</code>	Generates an expected (ideal) output signal used to compare with the actual output signal. You can specify one of the following values: <ul style="list-style-type: none"> <li>▪ <code>0</code> (default) does not generate an expected output.</li> <li>▪ <code>1</code> generates the expected output waveform (with a <code>#e</code> extension) based on the VEC/VCD file.</li> </ul>
<code>-node node_name {node_name}</code>	Multiple nodes are allowed with multiple node names. Without this option, <code>-print_expected</code> is a global option.
<code>-check_io message</code>	Checks if the IO statement is missing from the vector file. You can specify: <ul style="list-style-type: none"> <li>▪ <code>warn</code> (default) for the CustomSim tool to give a warning message and continue the simulation.</li> <li>▪ <code>error</code> for the CustomSim tool to issue an error message and stop the simulation.</li> </ul>

### Examples

```
set_vector_option -print_expected 1
```

Generates the expected output signal in the waveform file for visual comparison with the actual output signal waveform.

```
set_vector_option -print_expected 1 -node d[0]
```

Generates the expected output for the `d[0]` signal name.

```
set_vector_option -print_expected 1 -node d[0] d[1] d[2] s[0]
```

Generates the expected output for the following signal names: d[0], d[1], d[2], and s[0].

```
set_vector_option -check_z_state 1 -check_u_state 1
```

Instructs the CustomSim tool to report the mismatch of the high impedance state (Z) and U-state between simulated node states and the states specified in the VEC/VCD file.

---

## **set\_waveform\_option**

Sets the output waveform options. You can specify the file format, file-split size, voltage resolution, current resolution and flush percentage.

### **Syntax**

```
set_waveform_option [-grid_v grid_val]
[-grid_i grid_val]
[-compress_v compress_val]
[-compress_i compress_val]
[-tres time_resolution_value]
[-flush flush_time | flush_percentage_value%]
[-format format]
[-file split|merge]
[-size max_file_size_MB]
[-disk_full space_in_MB]
[-output_step step_size]
[-psf_lib_path absolute_path]
```

---

<b>Argument</b>	<b>Description</b>
-grid_v <i>grid_val</i>	Controls the grid size of the output voltage signal. All output points are rounded to the nearest integer of the grid. If set to 0, the rounding is turned off.

Argument	Description
<code>-grid_i grid_val</code>	Controls the grid size of the output current signal. All output points are rounded to the nearest integer of the grid. If set to 0, the rounding is turned off.
<code>-compress_v compress_val</code>	Controls the voltage compression. If set to 0, the compression is turned off.
<code>-compress_i compress_val</code>	Controls the current signal compression. If set to 0, the compression is turned off.
<code>-tres time_resolution_value</code>	Specifies waveform file minimum time resolution value by rounding. The default is 0.1p. If set 0, the rounding is turned off.
<code>-flush flush_time   flush_percentage_value%</code>	By default, the CustomSim tool writes to the output file when the output file buffer is full. This argument forces periodic writes to the output file. If a <i>flush_time</i> is specified, the output file is written at integer multiples of flush time. The flush time is in terms of the transient simulation time, not real time. The value is seconds. If a percentage is specified by using a % character, the file is written at integer multiples of the percentage of the simulation transient end time.
<code>-format [fsdb out wdf tr0 psfbin vpd utf]</code>	Sets the output waveform file format. See <a href="#">Supported Waveform Formats</a> in the <i>CustomSim User Guide</i> for more details. Note that the <code>utf</code> format is only available on the amd64 platform. For <code>fsdb</code> you can specify FSDB version 5.1 or 5.3. The default is 5.3.

Argument	Description
<code>-file [split   merge]</code>	If you specify the vpd output waveform file format, you can use the <code>-file</code> argument to set the file generation mode during CustomSim-VCS cosimulation. If you specify <code>split</code> , two vpd files are generated by the CustomSim tool and VCS separately. If you specify <code>merge</code> , one vpd file is generated.
<code>-size max_file_size_MB</code>	Sets the maximum file size before file splitting. See <a href="#">Table 19</a> .
<code>-disk_full space_in_MB</code>	The CustomSim tool periodically checks the remaining disk space in the output file disk. If the available disk space is less than twice the specified value, the CustomSim tool issues a warning. If you specify this argument and the available disk space is less than the specified value, and the CustomSim tool was invoked with the interactive mode command line option, <code>-intr</code> , the CustomSim tool enters interactive mode. Otherwise the CustomSim tool exits with an out of disk space error.  Note that if you do not specify this argument, the CustomSim tool only issues warnings.  The default value of this option is 100MB.

Argument	Description
<code>-output_step <i>step_size</i></code>	<p>When you set <code>-output_step</code> to a certain time value, such as <code>1ns</code> or <code>100ps</code>, the result format is in fixed time steps by every <code>1ns</code> or <code>100ps</code>, repetitively, to dump all probed waveform signals at every step. This capability works for all waveform formats.</p> <p>When you set both <code>-output_step</code> and <code>-tres</code>, the <code>-tres</code> argument is ignored and a warning message issued.</p> <p>Arguments such as <code>-compress_v/-compress/_i/-grid_v/-grid_i</code> are still honored, as long as the exact data point requested by <code>-output_step</code> is there.</p> <p>There is no default step size setting.</p>
<code>-psf_lib_path <i>absolute_path</i></code>	<p>To generate psf output for a CustomSim simulation, you need to have Cadence® IC 6.16.006 or later installed. Only the newer version of the Cadence tools has the necessary dynamic psf libraries that the CustomSim tool needs to generate psf output.</p> <p>There are two different ways to set the path to the psf libraries:</p> <ul style="list-style-type: none"> <li>▪ <code>set_waveform_option -psf_lib_path &lt;absolute path with leading character '/'&gt;</code></li> <li>▪ Set environment variable <code>CDSHOME &lt;home path with leading character '/'&gt;</code></li> </ul> <p>In addition to setting the path to the psf libraries, you also need to add the following command to your command file:</p> <pre>set_waveform_option -format psfbin</pre>

## Description

You can use the `set_waveform_option -format` switch to set the waveform output file format.

The order of precedence for specifying the waveform output file format is:

1. `-wavefmt` option on the command line
2. `set_waveform_option` command in a command script file (the command script file is evaluated as if the file were used on the final line of the netlist)
3. The last `.opt post=format` or `.opt xa_cmd="set_waveform_option . . . "` command that appears in the netlist

The `size` switch can be used to force the splitting of the waveform output file to a more convenient size. The value is specified in megabytes. The minimum file size for the formats that support file splitting is 10Meg. See [Table 19](#) for the CustomSim default resolutions and supported file-splitting features.

*Table 19 Supported file-splitting features (N/S means not supported, N/A means not applicable)*

Output Format	<code>-compress_v</code>	<code>-compress_i</code>	<code>-grid_v</code>	<code>-grid_i</code>	<code>-tres</code>	<code>-flush</code>	Split (-size)	32-bit <sup>1</sup>	64-bit <sup>1</sup>
fsdb	1uV	1pA	1uV	1pA	0.1p	Yes	Yes	2G	No split
out	1uV	1pA	1uV	1pA	0.1p	N/S	Yes	2G	No split
wdf	1uV	1pA	1uV	1pA	0.1p	Yes	Yes	2G	No split
tr0	1uV	1pA	N/S	N/S	0.1p	N/S	N/S	N/A	N/A
psfbin	1uV	1pA	N/S	N/S	0.1p	N/S	N/S	N/A	N/A
vpd	1uV	1pA	1uV	1pA	0.1p	Yes	Yes	2G	No split
utf	1uV	1pA	1uV	1pA	0.1p	Yes	Yes	2G	No split

1. Default automatic split sizes.

The `-compress_v` and `-compress_i` options are used to specify the resolution values for the slope-based lossy compression. Any data points that deviate from a straight line by less than this lossy compression value are dropped. The default voltage resolution is 1uV, and the default current resolution is 1pA.

The `-grid_v` and `-grid_i` options are used to specify the resolution values for rounding lossy compression. Any data points that deviate from resolution grid are rounded off to the closest grid. The default voltage resolution is 1uV, and the default current resolution is 1pA.

The `-flush` switch can be used to force more frequent writing to the output waveform file. Use this switch when you want to view the simulation progress during the simulation. If this command is not specified, the default setting is every 10%.

The disk full option alerts you that the output disk is nearing capacity. If the disk becomes completely full, unpredictable behavior may result. The `-disk_full` option can be used to raise warnings when the disk is nearing capacity (2x the set or default value) and to stop when the disk is nearly full (space is less than the set value). This may help to preserve the integrity of output files, but their integrity cannot be guaranteed.

If the `-disk_full` option is not specified the CustomSim tool only raises warnings when the disk has less than 200MB (2x the default value of 100MB). If this option is specified, the CustomSim tool stops with a disk full error if the disk space is detected to be less than the specified threshold.

The default value of disk full is 100MB. If the disk full option is not explicitly set the CustomSim tool starts to issue warnings when the disk has less than 200MB of space, but it does not stop with an error:

Warning: The available disk space 175MB is less than 200MB. Ctrl-C can suspend a simulation. Use the `set_waveform_option` command to adjust the disk full threshold.

If you specify:

```
set_waveform_option -disk_full 1000
```

The CustomSim tool issues a warning when the disk space is less than 2Gig and stops when it is less than 1G.

### Examples

```
set_waveform_option -format out -size 1000 -grid_i 1n -grid_v 1n  
-tres 1p -flush 5%
```

Sets the output format to `out`, a split size of 1000Meg, current grid to 1nA, voltage grid to 1nV, and writes to the waveform output file every 5% of the simulation time. The time resolution is set to a minimum of 1ps.

```
set_waveform_option -flush 100n
```

The previous example causes the CustomSim tool to update the waveform file after every 100ns of simulation time.

```
set_waveform_option -flush 150ns
```

The previous example causes the CustomSim tool to update the waveform file after every 150ns of simulation time.

```
set_waveform_option -flush 10%
```

The previous example causes the CustomSim tool to update the waveform file after every 10% of the simulation time. If the transient end time is 2us, in this example the CustomSim tool updates the waveform file after at least every 200ns.

### See Also

[set\\_probe\\_window](#), [enable\\_print\\_statement](#)

[Supported Waveform Formats](#) in “Running the Simulator” in the *CustomSim User Guide*.

---

## **set\_waveform\_sim\_stat**

Lets you display the CPU (or wall) time as a function of the transient simulation time in the same WaveView window. This command helps you detect bottlenecks when the CPU or wall time is increasing faster than the transient simulation time.

### Syntax

```
set_waveform_sim_stat -type sim_stat_type {sim_stat_type}
```

Argument	Description
<code>-type sim_stat_type {sim_stat_type}</code>	<p>Specify one or more of the following options:</p> <ul style="list-style-type: none"> <li>▪ <code>cpu</code> dumps the CPU time versus the transient simulation time in the waveform file.</li> <li>▪ <code>wall</code> dumps the wall time versus the transient simulation time in the waveform file.</li> <li>▪ <code>vm</code> dumps the virtual memory usage versus the transient simulation time in the waveform file.</li> <li>▪ <code>pvm</code> dumps the peak virtual memory usage versus the transient simulation time in the waveform file.</li> <li>▪ <code>pm</code> dumps the peak physical memory (which resides in core memory) versus the transient simulation time in the waveform file.</li> <li>▪ <code>all</code> dumps the CPU and wall times and memory usage versus transient simulation time in the waveform file. It also dumps <code>cpu_load</code> and remaining time versus transient simulation time.</li> <li>▪ <code>cpu_load</code> dumps the CPU load in the waveform file</li> <li>▪ <code>remaining</code> dumps the estimated remaining time (as shown on stdout) is in the waveform file.</li> </ul>

### Description

The `set_waveform_sim_stat` command dumps the specified time type (either CPU time, wall time, or both, or memory) in the waveform output file on the Y axis (X axis being the usual transient simulation time). The signal labels on the Y axis are CPU Time, Wall Time, Virtual Mem, Peak Virtual Mem and Physical Memory. The Y-axis unit is seconds for all the times and is MB for the memory. This new wave is reported in the waveform output file in all the supported output file formats.

The CPU or wall time or memory value is reported every time it is updated in the standard output (or in the GRID or LSF log file).

You can specify multiple `set_waveform_sim_stat` commands.

### Examples

```
set_waveform_sim_stat -type wall cpu pvm
```

Dumps the wall, CPU, and peak virtual memory as f(transient time) in the log file and as Y-axis values, while the X-axis shows the transient time.

```
set_waveform_sim_state -type wall
```

Dumps the wall time=f(transient time) in the log file.

---

## set\_wildcard\_rule

Enables the CustomSim tool to match *all* levels of the hierarchy—or only *one* level—depending on your specifications. By default, the CustomSim tool follows HSPICE behavior, which matches all levels of the hierarchy.

### Syntax

```
set_wildcard_rule -match* all | one  
[-node_alias enable_value]  
[-match_ic* all|one]
```

Argument	Description
-match* all one	Specifies * to match any number of hierarchy levels (default) or one level of hierarchy.
-node_alias enable_value	When you enable this option all node aliases are reported in waveform files and the <a href="#">report_node_cap</a> command also reports the capacitance information for all node aliases. The default is 0.
-match_ic* all one	Specify * or all to match any number of hierarchy levels (default), or one to specify one level of hierarchy for .ic statements.  If you specify a value for this option it overrides the value set by the -match* option for .ic statements. Other statements with wildcards, such as .probe, are not affected by the -matchic* option.

---

### Description

Sets rules for wildcard \* matching in instance and node hierarchical names.

If the `one` argument is set, the CustomSim tool matches the asterisk ( `*` ) character to one level of the hierarchy; otherwise, all levels of the hierarchy are matched.

### Examples

```
set_wildcard_rule -match* one
```

### See Also

[probe\\_waveform\\_current](#), [probe\\_waveform\\_logic](#),  
[probe\\_waveform\\_voltage](#), [report\\_power](#)

---

## set\_zstate\_option

Sets the conducting rules for the [check\\_node\\_zstate](#) command.

### Syntax

```
set_zstate_option -idsth value_1
                  [-vbeth value_2]
                  [-rule rule_value {rule_value}]
                  [-diode_vth value]
                  [-va_rule value]
                  [-xdummy value]
```

---

Argument	Description
<code>-idsth value_1</code>	Specifies the threshold for a MOSFET to be considered conducting when <code>ids</code> exceeds <code>value_1</code> . If not specified, the default is <code>1e-8</code> ampere. See the Description section for more information about the conducting criteria.
<code>-vbeth value_2</code>	Specifies the threshold for BJT to be considered conducting when <code>Vbe</code> exceeds <code>value_2</code> . If not specified, the default is <code>0.64</code> volt. See the Description section for more information about the conducting criteria.

Argument	Description
<code>-rule rule_value {rule_value}</code>	Selects the rule of conducting for a MOSFET. The MOSFET is considered conducting if one of the rules is met. Other rules are as follows:  NMOS: <ul style="list-style-type: none"> <li>▪ <math>V_{gs} &gt; V_{th}</math> (rule=1)</li> <li>▪ <math>I_{ds} &gt; idsth</math> (rule=2)</li> <li>▪ <math>V_g &gt; VDD - 0.1</math> (rule=3)</li> </ul> PMOS: <ul style="list-style-type: none"> <li>▪ <math>V_{gs} &lt; V_{th}</math> (rule=1)</li> <li>▪ <math>I_{ds} &gt; idsth</math> (rule=2)</li> <li>▪ <math>V_g &lt; 0.1</math> (rule=3)</li> </ul> The default is 1 and 2.
<code>-diode_vth value</code>	Specifies the threshold for a diode to be considered conducting when $V_{ac}$ exceeds the specified value. If you specify a value less than 0, diode is considered as not conducting. The default is 0.2V.
<code>-va_rule value</code>	By default, or when this argument is set to 0, the output nodes of Verilog-A modules are reported if there is no conducting path from the voltage source to the output nodes. When this argument is set to 1, those nodes are not reported in the output file.
<code>-xdummy value</code>	Enables additional checking on the fanout devices (MOS and BJT) of the HiZ node. A MOS is a dummy device if and only if: (source && gate are in HiZ) or (drain && gate are in HiZ). BJT is a dummy device if and only if: (collector && base are in HiZ) or (emitter && base are in HiZ).  Set this argument to 0 (default) to disable it or to 1 to enable the check. A HiZ node is reported only when it connects to at least one non-dummy transistor (MOS or BJT). For a node that is HiZ, fanout devices are also reported along with the HiZ node.

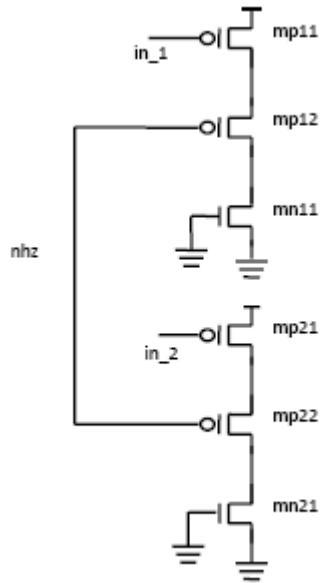
### Description

This command specifies the conducting rules so that the CustomSim tool can diagnose specified nodes staying in a high-impedance (floating) state for a specific period of time. The detected nodes are reported to an error file with suffix of `.errz`.

A node stays in a high-impedance state if there is no conducting path from any voltage source to the node. A conducting path consists of conducting elements. An element is conducting if that specific element meets the following criteria:

Device	Rule
NMOS	$V_{gs} > V_{th}$ (rule=1)    $I_{ds} > idsth$ (rule=2)    $V_g > VDD - 0.1$ (rule=3) The default for $idsth$ is $1e-8A$ . The default rule is 1 & 2.
PMOS	$V_{gs} < V_{th}$ (rule=1)    $I_{ds} > idsth$ (rule=2)    $V_g < 0.1$ (rule=3) The default for $idsth$ is $1e-8A$ . The default rule is 1 & 2.
NPN	$V_{be} > V_{beth}$ The default for $V_{beth}$ is 0.64
PNP	$V_{be} < -V_{beth}$ The default for $V_{beth}$ is 0.64
Diode	$I(diode) > diode\_ith$    $V(a,c) > diode\_vth$ The default for $diode\_ith$ is $1e-8A$ .
Resistor	Always conducting by default.
Inductor	Always conducting.
Capacitor	Never conducting.
Verilog-A	Never conducting.
Controlled sources	Always conducting.

## Examples



In the previous example, node nhz is a HiZ node. It is always reported by default or when -xdummy is set to 0. When -xdummy is set to 1, it enables the check of the fanout devices of the node nhz. If the fanout devices are off, then node nhz is not reported. In this case, node nhz is not reported if mp12 and mp22 are both off (dummy). The dummy states of mp12 and mp22 are controlled by node in\_1 and in\_2. When the controlling node is logical high, the transistor being controlled is dummy. For those that remain as HiZ, the fanout transistors are reported along with the HiZ node.

---

## skip\_circuit\_block

Provides the equivalent effect of commenting out instances without editing netlist files. For subcircuit instances, it can also estimate their loads and replace them with capacitive loads at the ports.

### Syntax

```
skip_circuit_block [-load] enable_value instance_spec
```

## Description

Be careful how you use this command, because it can affect the circuit connectivity. It checks for floating nodes, gates, and dangling nodes after the circuit block has been removed and issues appropriate warnings.

The `-load` argument turns on and off instance-based load estimation. The default is 0. If you specify `-load 1`, instead of commenting out the specified instances, they are replaced with capacitive loads at the ports.

## Examples

### *Example 52*

```
skip_circuit_block -inst x1.xregulator
```

Removes the `x1.xregulator` instance from the netlist.

### *Example 53*

```
skip_circuit_block -subckt pdetect
```

Removes all instances of the `pdetect` subcircuit from the netlist.

### *Example 54*

```
skip_circuit_block -inst d1 -subckt nmosmac
```

Removes `d1` from all instances of the `nmosmac` subcircuit.

### *Example 55*

```
skip_circuit_block -load 1 -inst x1.xregulator
```

Replaces the `x1.xregulator` instance with capacitive loads at the ports.

### *Example 56*

```
skip_circuit_block -load 1 -subckt pdetect
```

Replaces all instances of the `pdetect` subcircuit with capacitive loads at the ports.

---

## source

`source` is a native TCL command, not a CustomSim-specific command. You use it to include other command script files.

**Syntax**

```
source file_name
```

**Description**

This command reads and references another command script file. Use it with the `-c` command line option, not the `.opt xa_cmd` option.

**Examples***Example 57*

```
#Test case-specific file
set_sim_level 4
#Load common settings
source xa_common_settings.tcl
```

## CustomSim Interactive Command Syntax

---

*Provides the syntax for all of the CustomSim interactive and DC interactive mode commands.*

The CustomSim interactive commands are:

```
alias
icheck_node_zstate
iclose_log
icontinue_sim
idelete_break_point
iforce_node_voltage
ilist_break_point
ilist_force_node
imatch_elem
imatch_node
iopen_log
iprint_connectivity
iprint_dcpath
iprint_elem_info
iprint_exi
iprint_flash_cell
iprint_help
iprint_node_info
iprint_subckt
iprint_time
iprint_tree
iprobe_waveform_current
iprobe_waveform_voltage
iquit_sim
irelease_node_voltage
ireport_node_cap
```

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

```
ireport_operating_point  
isearch_node  
iset_break_point  
iset_diagnostic_option  
iset_interactive_option  
iset_interactive_stop  
iset_save_state  
iset_zstate_option
```

The DC interactive mode commands are:

```
iclose_log  
icontinue_dc  
idelete_node_ic  
imatch_elem  
imatch_node  
iopen_log  
iprint_connectivity  
iprint_elem_info  
iprint_exi  
iprint_help  
iprint_node_info  
isearch_node  
iset_node_ic
```

---

## CustomSim Interactive Commands

This section describes the interactive commands:

- [alias](#)
- [icheck\\_node\\_zstate](#)
- [iclose\\_log](#)
- [icontinue\\_sim](#)
- [idelete\\_break\\_point](#)
- [iforce\\_node\\_voltage](#)
- [ilist\\_break\\_point](#)
- [ilist\\_force\\_node](#)

- [imatch\\_elem](#)
- [imatch\\_node](#)
- [iopen\\_log](#)
- [iprint\\_connectivity](#)
- [iprint\\_dcpath](#)
- [iprint\\_elem\\_info](#)
- [iprint\\_exi](#)
- [iprint\\_flash\\_cell](#)
- [iprint\\_help](#)
- [iprint\\_node\\_info](#)
- [iprint\\_subckt](#)
- [iprint\\_time](#)
- [iprint\\_tree](#)
- [iprobe\\_waveform\\_current](#)
- [iprobe\\_waveform\\_voltage](#)
- [iquit\\_sim](#)
- [irelease\\_node\\_voltage](#)
- [ireport\\_node\\_cap](#)
- [ireport\\_operating\\_point](#)
- [isearch\\_node](#)
- [iset\\_break\\_point](#)
- [iset\\_diagnostic\\_option](#)
- [iset\\_env](#)
- [iset\\_interactive\\_option](#)
- [iset\\_interactive\\_stop](#)
- [iset\\_save\\_state](#)
- [iset\\_zstate\\_option](#)

## alias

Creates an alias name for the interactive commands. When arguments are specified, an alias is defined for each *alias\_name* for whose *actual\_name* is given.

### Syntax

```
alias alias_name actual_name
```

---

Argument	Description
<i>alias_name</i>	Creates an alias for an interactive command.
<i>actual_name</i>	Defines the interactive command.

---

### Examples

#### *Example 58*

```
alias pns iprint_node_info
```

[Example 58](#) creates an alias *pns* for the *iprint\_node\_info* command.

#### *Example 59*

```
alias ipe iprint_elem_info -index
```

[Example 59](#) creates an alias *ipe* for the *iprint\_elem\_info* command with the argument *-index*.

---

## icheck\_node\_zstate

Performs a high-impedance node check in interactive mode.

### Syntax

```
icheck_node_zstate -node node_name {node_name}  
[-fanout <0|1|2>]  
[-rule rule_value {rule_value}]  
[-subckt subckt_name {subckt_name}]  
[-except_subckt subckt_name {subckt_name}]  
[-diode_vth value]  
[-idsth ids_value]  
[-vbeth vbeth_value]
```

```
[-except_node node {node}]
[-file file_name]
[-report report_value {report_value}]
```

Argument	Description
-node node_name {node_name}	Specifies the node names at which the high-impedance state is checked. The node name can be a single node or a node name with a wildcard character that represents a group of node names. The rule of asterisk (*) character is controlled by <a href="#">setWildcardRule</a> .
-fanout <0 1 2>	Selects the type of nodes for high-impedance checking: <ul style="list-style-type: none"> <li>▪ A value of 0 enables all nodes.</li> <li>▪ A value of 1 enables nodes that have a direct connection to a transistor gate.</li> <li>▪ A value of 2 enables all nodes that have a direct connection to a transistor bulk.</li> </ul> If you do not use this option, the default is 0.
-rule rule_value {rule_value}	Selects the rule of conducting for a MOSFET. The MOSFET is considered conducting if one of the rules is met. Other rules are as follows: <p>NMOS:</p> <ul style="list-style-type: none"> <li>▪ <math>V_{GS} &gt; V_{TH}</math> (-rule 1)</li> <li>▪ <math>I_{DS} &gt; I_{DSTH}</math> (-rule 2)</li> <li>▪ <math>V_G &gt; V_{DD} - 0.1</math> (-rule 3)</li> </ul> <p>PMOS:</p> <ul style="list-style-type: none"> <li>▪ <math>V_{GS} &lt; V_{TH}</math> (-rule 1)</li> <li>▪ <math>I_{DS} &gt; I_{DSTH}</math> (-rule 2)</li> <li>▪ <math>V_G &lt; 0.1</math> (-rule 3)</li> </ul> The default is 1 and 2.
-subckt subckt_name {subckt_name}	Specifies the subcircuits to be checked. Wildcard characters are allowed in subcircuit names.
-except_subckt subckt_name {subckt_name}	Specifies the subcircuits to be excluded from high-impedance checking. This argument should be used only when <code>-node node_name</code> contains wildcard characters. Wildcard characters are allowed in subcircuit names

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

Argument	Description
<code>-diode_vth value</code>	Sets the forward bias threshold for diodes. A diode with v(a,c) greater than <i>value</i> is considered conducting for a high impedance check. A value less than 0 causes diodes to be always considered non-conducting. The default value is 0.2V.
<code>-idsth ids_value</code>	Specifies the threshold for a MOSFET to be considered conducting when ids exceeds <i>value_1</i> . If not specified, the default is 1e-8 ampere. See the Description section for more information about the conducting criteria.
<code>-vbeth vbeth_value</code>	Specifies the threshold for BJT to be considered conducting when Vbe exceeds <i>value_2</i> . If not specified, the default is 0.64 volt. See the Description section for more information about the conducting criteria.
<code>-except_node node_name {node_name}</code>	Specifies the nodes to be excluded from high-impedance checking. This argument should be used only when <code>-node node_name</code> contains wildcard characters.
<code>-file file_name</code>	If you specify a file name the output is written to that file instead of standard output. If you specify an existing file name, that file is overwritten.
<code>-report report_value {report_value}</code>	You can specify the following keywords: <ul style="list-style-type: none"><li>▪ <code>node</code> lists Hi-z nodes.</li><li>▪ <code>channel</code> lists all channel-connected devices for the node.</li><li>▪ <code>gate</code> lists all gate-connected devices for the node.</li></ul> The default is <code>node</code> .

**Description**

This interactive command enables the CustomSim tool to diagnose specified nodes staying in a high-impedance (floating) state.

A node stays in a high-impedance state if there is no conducting path from any voltage source to the node. A conducting path consists of conducting elements. An element is conducting if that specific element meets the following criteria:

Device	Rule
NMOS	<ul style="list-style-type: none"> <li>▪ <math>V_{gs} &gt; V_{th}</math> (rule=1)</li> <li>▪ <math>I_{ds} &gt; idsth</math> (rule=2)</li> <li>▪ <math>V_g &gt; VDD - 0.1</math> (rule=3)</li> </ul>
PMOS	<ul style="list-style-type: none"> <li>▪ <math>V_{gs} &lt; V_{th}</math> (rule=1)</li> <li>▪ <math>I_{ds} &gt; idsth</math> (rule=2)</li> <li>▪ <math>V_g &lt; 0.1</math> (rule=3)</li> </ul> <p>For more information about the rule values, see <a href="#">iset_zstate_option</a>.</p>
NPN	$V_{be} > v_{beth}$
PNP	$V_{be} < -v_{beth}$
Diode	Forward-biased.
Resistor	Always conducting.
Inductor	Always conducting.
Capacitor	Never conducting.
Verilog-A	Never conducting.
Controlled sources	Always conducting.

To diagnose if a high impedance condition is harmful, you need to know which gates it is driving. To diagnose how to fix the high impedance condition, the channel-connected elements must be reported. The voltage on the high impedance nodes is always reported. The hierarchical subcircuit names are always be printed when you specify  
`-report gate and channel`. The MOS terminal voltages are also printed.

---

## **iclose\_log**

Closes the interactive mode log file that was opened by the [iopen\\_log](#) command. The log file contains all records of interactive mode commands and

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

the results reported by the commands entered between `iopen_log` and `iclose_log`.

**Syntax**`iclose_log`**See Also**[iopen\\_log](#)

---

**icontinue\_sim**

Continues the transient simulation from the initial stop point to the following stop point. If `time` and `unit` are specified, the simulation stops and enters the interactive mode at time  $t + (\text{time}) (\text{unit})$ . The `-to` option specifies the absolute time to which the simulation proceeds, then enters interactive mode.

**Syntax**`icontinue_sim -i time[unit] [-to time[unit]]`

---

Argument	Description
<code>-i time[unit]</code>	<p>Stops the simulation and enters the interactive mode at time <math>t + (\text{time}) (\text{unit})</math></p> <p><math>t</math> is the current simulation time.</p> <p>If <math>t + (\text{time}) (\text{unit})</math> is less than the simulation stop time, the simulation continues until <math>t + (\text{time}) (\text{unit})</math>, then enters the interactive mode at <math>t + (\text{time}) (\text{unit})</math>.</p> <p>White space is not allowed between <code>time</code> and <code>unit</code>.</p> <p>If <code>time [unit]</code> is not specified, the simulation continues until it is completed, or until the <code>ctrl-C</code> is entered.</p>
<code>-to time[unit]</code>	The default for <code>unit</code> is seconds, if not specified.
<code>-to time[unit]</code>	Specifies the absolute time to which the simulation proceeds, then enters interactive mode. For example, if you specify <code>icontinue_sim -to 11ns</code> at time 9ns, the transient simulation runs to 11ns. This command issues a warning, and is ignored, if the time specified by <code>-to</code> is less than the current time.

---

## Examples

```
icontinue_sim 10n
```

This example continues the simulation for another 10ns.

```
icontinue_sim -to 10n
```

This example runs the simulation to 10ns and returns to interactive mode, assuming current time is less than 10ns.

---

## idelete\_break\_point

Removes the stop points.

### Syntax

```
idelete_break_point -point all|stop_point1  
[stop_point2 ... stop_pointn]
```

---

Argument	Description
-point all	Removes all the stop points.
-point stop_point1 stop_point2... stop_pointn	Removes the specified stop points.

---

## Examples

```
XA> ilist_break_point
```

```
1: break at time: 1 ns  
2: break at time: 10 ns
```

```
XA> idelete_break_point 2  
XA> ilist_break_point
```

```
1: break at time: 1 ns
```

### Example 60

```
idelete_break_point -point 2
```

**Example 60** removes stop point 2.

### See Also

[ilist\\_break\\_point](#)

---

## **iforce\_node\_voltage**

Forces the specified nodes to stay at the specified constant voltage. The node voltage stays at the same value from the current time until either the end of simulation or when the constant node voltage status is released by [irelease\\_node\\_voltage](#).

### **Syntax**

```
iforce_node_voltage -node node_name {node_name}  
    -v[oltage] voltage_value  
    [-slope t_value]  
    [-time time_val]
```

---

Argument	Description
-node <i>node_name</i> { <i>node_name</i> }	Specifies the node name to force.
-voltage <i>voltage_value</i>	Specifies the voltage value to set.
-slope <i>t_value</i>	Forces the voltage with a ramp of <i>t_value</i> (in seconds per volt). The default is 1ps and must be positive a positive value.
-time <i>time_val</i>	Specifies the time when the nodes are forced. The default is the current break point.

---

### **Examples**

```
XA> iforce_node_voltage -node cn -voltage 2
```

The *cn* node is forced at 2 V starting at the current time. It remains at this value until the end of the simulation unless you use [irelease\\_node\\_voltage](#) command for the same node.

---

## **ilist\_break\_point**

Lists the existing stop points.

### **Syntax**

```
ilist_break_point -list [number]
```

---

Argument	Description
-list <i>number</i>	Lists the last <i>number</i> of the stop points. If not specified, lists all the stop points.

---

### Examples

```
XA> ilist_break_point -at 1n
XA> iset_break_point -at 10n
XA> ilist_break_point

1: break at time: 1 ns
2: break at time: 10 ns
```

#### *Example 61*

`ilist_break_point -list`

[Example 61](#) lists the existing stop points.

---

## **ilist\_force\_node**

Lists all nodes you specified with [force\\_node\\_voltage](#).

### Syntax

`ilist_force_node -file file_name`

---

Argument	Description
-file <i>file_name</i>	Specifies the file name that contains the list of forced nodes.

---

### Description

If you specify the `-file` argument, the forced node list is written to the specified file. Otherwise the list is echoed to the standard output.

Note that a node forced with [force\\_node\\_voltage](#) is not listed in the output of this command unless the simulation time advanced since the node was forced.

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

**Examples**

```
XA> iforce_node_voltage xbuffer.pd_d -v 1.2
XA> iforce_node_voltage xbuffer.sigi -v 0
XA> ilist_force_node

XA> icont ln
XA> ilist_force_node
xbuffer.pd_d (4) = 1.2
xbuffer.sigi (3) = 0
```

---

**imatch\_elem**

Prints a list of the element indexes and hierarchical element names that match the specified pattern.

**Syntax**

```
imatch_elem -pattern pattern ...
```

Argument	Description
<i>pattern</i>	Patterns can contain the * wildcard character. The printed list shows all matches to the specified pattern. You can set wildcard matching for the hierarchy with <a href="#">iset_interactive_option</a> .

**Examples**

```
XA> imatch_elem *x1*
5 x1.r1
6 x1.r2
7 x1.x1.r1
```

In this example, the \*x1\* pattern matches 3 elements.

**See Also**

[iset\\_interactive\\_option](#)

---

**imatch\_node**

Prints a list of the node indexes and node names that match the specified pattern.

## Syntax

```
imatch_node -pattern pattern ...
  [-limit level]
  [-port enable_value]
```

---

Argument	Description
<i>pattern</i>	Patterns can contain the * wildcard character. The printed list shows all matches to the specified pattern. You can set wildcard matching for the hierarchy with <a href="#">iset_interactive_option</a> .
-limit <i>level</i>	Specifies wildcard matching only down to the specified hierarchical depth. The default hierarchical limit level is 3.
-port <i>enable_value</i>	Enables or disables the matching of subcircuit port names. The default value of is taken from the <a href="#">iset_interactive_option</a> command.

---

## Description

imatch\_node prints a list of matched nodes. The CustomSim tool first reports the value of -limit applied, then lists the node index and node name matched with one item per line. Finally, the CustomSim tool reports the total number of matched nodes.

The [iset\\_interactive\\_option](#) command settings apply to this command. The \*\* wildcard can be set to match or not to match the hierarchical delimiter. The \*\* wildcard \* only matches primary node names, unless you specify the -port argument. When you specify -port, the \*\* wildcard also matches alias node names.

## Examples

```
XA> imatch_node *a
1 a
2 x1.a
3 x1.fa
pattern *a matched 3 nodes
```

In this example, the \*a pattern matches 3 nodes.

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

```
XA> imatch_node *a -limit 0
imatch_node using -limit 0
1 a
pattern *a matched 1 nodes
```

This example finds all nodes ending in `a` at the top-level of the netlist, level 0.

```
XA> imatch_node *out -limit 4 -port 1

15 x0.x1.x2.aout
15 x0.x1.x2.x3a.out
16 x0.x1.x2.bout
16 x0.x1.x2.x3b.out
```

This example finds all nodes ending in `out` down to the hierarchical depth of 4 and also reports ports.

**See Also**

[iset\\_interactive\\_option](#)

---

## iopen\_log

Opens the interactive mode `logfile_name`, which contains the record of the interactive mode commands and the results reported by these commands until the log file is closed by the [iclose\\_log](#) command. Only one log file can be opened at one time.

**Syntax**

```
iopen_log -file logfile_name [-mode append|write]
```

---

Argument	Description
<code>-file <i>logfile_name</i></code>	Specifies the interactive mode log file name.
<code>-mode append</code>	Specifies that contents are appended to pre-existing files using the same file name.
<code>-mode write</code>	Specifies that contents overwrite pre-existing files using the same file name. Default is <code>-mode write</code> .

---

**Examples**

```
iopen_log -file logfile
```

This example opens the log file named `logfile`.

### See Also

[iclose\\_log](#)

## iprint\_connectivity

Prints the detailed node connectivity information for the given node names or indices. The elements are categorized into channel-connected, gate-connected, and other elements.

### Syntax

```
iprint_connectivity -node node_name {node_name}
[-print gc|cc|o|all]
[-on current_value]
[-file file_name]
[-file_append file_name]
```

or

```
iprint_connectivity -index node_index {node_index}
[-print gc|cc|o|all]
[-on current_value]
[-file file_name]
[-file_append file_name]
```

Argument	Description
<code>-node <i>node_name</i> {<i>node_name</i>}</code>	Specifies the node name, which can contain wildcard characters.
<code>-print gc cc o all</code>	Specifies to selectively print the gate-connected, channel-connected, other, or all elements. If not specified, the default prints gate-connected and channel-connected elements.
<code>-index <i>node_index</i></code>	Specifies node index.
<code>-on <i>current_value</i></code>	Prints only those instances that have a current exceeding the specified <i>current_value</i> .

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

---

<b>Argument</b>	<b>Description</b>
<code>-file <i>file_name</i></code>	Writes output only to the specified file. If the file exists it is overwritten. The file is located in the directory specified by the <code>-o</code> command line argument.
<code>-file_append <i>file_name</i></code>	Writes output only to the specified file. If the file exists this argument appends the output to the existing file, otherwise it creates the specified file. The file is located in the directory specified by the <code>-o</code> command line argument.

---

**Examples**

```
iprint_connectivity -node xspine_0.xdqr_0.xl850.osc_2
iprint_connectivity xspine_0.xdqr_0.xl850.osc_2 -print cc
iprint_connectivity -index 3
```

**See Also**

[iprint\\_node\\_info](#)

**iprint\_dcpath**

Finds and prints the DC path information.

**Syntax**

```
iprint_dcpath -ith ival [-node node_name {node_name}]
[-at tval {tval}]
[-file file_name]
```

or

```
iprint_dcpath -ith ival [-node node_name {node_name}]
[-period period_value]
[-start start_time]
[-end end_time]
[-file file_name]
```

---

<b>Argument</b>	<b>Description</b>
<code>ith <i>ival</i></code>	Sets the current threshold value. The default is 50uA.

---

---

Argument	Description
<code>-node node_name {node_name}</code>	Specifies the terminal node names of the DC current path. The DC path search starts from any node specified in this node list and ends when it reaches either another node in the list or a DC voltage source node. If you do not specify a node, the CustomSim tool reports DC current paths between any pair of voltage source nodes.
<code>-at tval {tval}</code>	Specifies the specific time points at which the path checking occurs.
<code>-period period_value</code>	When you specify the <code>-period</code> argument the DC path search occurs at integer multiples of <code>period_value</code> beginning at the current time if you do not use the <code>-start</code> argument.
<code>-start start_time</code>	Specifies the first time that periodic path checking occurs. If you do not use <code>-start</code> periodic checking starts at the current time.  You can only use <code>-start</code> with the <code>-period</code> argument.
<code>-end end_time</code>	Specifies that no path checking occurs past the <code>end_time</code> . If you do not use <code>-end</code> path checking occurs until the end of the simulation.  You can only use <code>-end</code> with the <code>-period</code> argument.
<code>-file file_name</code>	Writes the DC path report to the specified file.

---

### Description

The `iprint_dcpath` command searches for and reports DC paths. The DC path search starts from any specified node and ends when the search reaches either another node in the list or a DC voltage source node. If you do not specify a node, the CustomSim tool reports the DC current paths between any pair of voltage source nodes. The path is only reported through MOS and resistor elements.

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

**Examples**

```
XA> iprint_dcpath -ith 1e-6  
  
path 1 from vdd to 0 @ 1.04e-07s  
  
xmos.mn (NMOS: lds=3.33804e-06  
    drain vdd 3  
    source xmos.n1 0.0530892  
    gate g 0.6  
xmos.mn2 (NMOS: lds=3.33793e-06  
    source 0 0  
    drain xmos.n1 0.0530892
```

---

**iprint\_elem\_info**

Prints the detailed element information for the given element names at the specific time of activation.

**Syntax**

```
iprint_elem_info -elem element_name {element_name}  
    [-report brief]  
    [-file file_name]
```

or

```
iprint_elem_info -index elem_index {elem_index}  
    [-report brief]  
    [-file file_name]
```

---

Argument	Description
-elem element_name	Specifies the element name, which can include wildcard characters.
-index elem_indexn	Specifies element index.
-report brief	Provides a brief version of the output that reports only the element terminal connectivity, voltage, and currents. Other information is suppressed.
-file file_name	Writes iprint_elem_info output to the specified file.

---

## Description

You can also provide a subcircuit instance as an element. In this case, the CustomSim tool prints the subcircuit name, the list of its ports, and the voltages on each port. The detailed element information includes:

- Element name, element type, and model name
- Element terminal connectivity
- Element parameters and values
- Element terminal voltages

MOSFET-specific information:

- MOS logic state (ON/OFF)
- MOS effective length and width ( $L_{eff}$  and  $W_{eff}$ )
- MOS conductance ( $g_{ds}$  and  $g_m$ )
- MOS threshold voltage ( $V_{th}$ )
- MOS Voltage-dependent diode capacitance ( $c_{bs}$  and  $c_{bd}$ )
- MOS Voltage-dependent gate capacitance ( $c_{gs}$ ,  $c_{gb}$ , and  $c_{gd}$ )
- MOS  $I_{ds}$  current ( $I_{ds}$ )

## Examples

```
XA> iprint_elem_info x1.xi@1477.mpt1

Elem=X1.XI@1477.MPT1 (203) Type=PMOS Model=PENH
D=X1.X001 (232) G=X1.N@1514 (130) S=VDD (14) B=VDD (14)

Vd=0.0108579 Vg=1.69226 Vs=1.69213 Vb=1.69213
Weff=11.9178u Leff=0.205123u PD=13.4u PS=13.4u AD=4.2u^2 AS=4.2u^2
Vt=-0.73418 OFF
Ids=-2.7808e-07u
gds=2.86161e-13 gm=6.111e-12
cgs=3.03919f cgd=3.03903f cgb=4.59196f cbs=6.99465f cbd=4.48262f
id=0.105008u ig=-0.0446823u is=0.000726083u ib=-0.0610515u
```

```
XA> iprint_elem_info X1.RI@5234
```

```
Elem=X1.RI@5234 (161) Type=R
N1=X1.N@5446 (195) N2=X1.N@5445 (194)
R=33
Vn1=1.69267 Vn2=1.69267
I=0.0036124u
```

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

*Example 62*

```
iprint_elem_info -elem x1.x2.m1
```

[Example 62](#) prints information for element x1.x2.m1.

*Example 63*

```
iprint_elem_info -index 1 3 2
```

[Example 63](#) prints information for element indices 1, 3, and 2.

*Example 64*

```
XA>iprint_elem_info x1
Elem=x1 Type=subckt subckt=mysub1
PORT 1 port=a - 11 (1) v=1
PORT 2 port=b - 12 (2) v=1
PORT 3 port=c - 13 (3) v=2.65
PORT 4 port=d - 14 (4) v=1
```

[Example 64](#) prints subcircuit information and port voltages for a subcircuit instance.

**See Also**

[iprint\\_connectivity](#), [iprint\\_node\\_info](#)

---

**iprint\_exi**

Prints elements with excessive currents.

**Syntax**

```
iprint_exi -inst inst_name {inst_name}
[-ith ivalue]
[-file file_name]
[-report report_value {report_value}]
```

---

Argument	Description
-inst <i>inst_name</i>	Specifies the hierarchical element names to be searched. You can use wildcard characters (*) in the instance names. The <a href="#">set_wildcard_rule</a> conventions apply.

---

Argument	Description
<code>-ith <i>ivalue</i></code>	Specifies the current threshold that must be exceeded for the element to be reported. The default value is 50uA.
<code>-file <i>file_name</i></code>	Specifies the output file name.
<code>-report <i>report_value</i> <i>{report_value}</i></code>	<i>report_value</i> can be: <ul style="list-style-type: none"> <li>▪ subname to specify a list of subcircuit names of the hierarchical instance components to be printed in the output.</li> <li>▪ brief to report only element terminal connectivity, voltages, and currents. Other information is suppressed.</li> </ul>

---

### Description

`iprint_exi` reports the current through any device terminal that exceeds the threshold. The following device types are checked and reported:

- MOS
- Resistor
- BJT
- Diode

### Examples

The output format for elements found that exceed the current threshold is the same as [iprint\\_elem\\_info](#). If you specify a list of subcircuit names with the `-report` argument, the hierarchical element instance names have the following general format:

`Elem=X0.X1.X2...Xn.modelname`

`Subckt: X0=x0name X1=X1name X2=x2name ... Xn=Xnname`

For example:

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

```
XA> iprint_exi -ith 1n xmos.m* -report subname
Elem=Xmos.mn (6) Type=NMOS Model=nch.7
D=vdd (5) G=g (3) S=Xmos.n1 (16) B=vb (4)
Vd=3 Vg=1.81293e-90 Vs=-0.00837316 Vb=1.81293e-90
M=1
Weff=2.016u Leff=0.948223u PD=2.365u PS=2.365u AD=0.351792u^2
AS=0.351792u^2 SA=0u SB=0u
Vt=0.19898 OFF
Ids=0.26352u
gds=5.38557e-10 gm=0
cgs=4.30157f cgd=0.416708f cgb=3.65854f cbs=1.20401f
cbd=0.487088f
id=0.26352u ig=-0.0663667u is=-0.00752308u ib=-0.189631u
Subckt: Xmos=mymos
```

---

**iprint\_flash\_cell**

Prints information for flash core cell elements.

**Syntax**

```
iprint_flash_cell -dvth value -inst inst_name
    {inst_name}
    [-file filename]
    [-save save_filename]
```

---

Argument	Description
-dvth <i>value</i>	Specifies the minimum vth change for cells to be printed.
-inst <i>inst_name</i>	Specifies the instance names. You can use an asterisk (*) wildcard character to represent all instances matching the pattern.
-file <i>filename</i>	Specifies a file name to which the command output is written.
-save <i>save_filename</i>	Specifies a file name to which a command file is written with the configuration command to initialize the cells to their current state. You can include this file to run subsequent simulations.

---

## Description

`iprint_flash_cell` identifies and reports the flash core cell instances from specified instance names having a threshold voltage shift (-dvth) with either of the following parameters:

- Greater than or equal to the specified value if it is positive or zero.
- Less than or equal to the specified value if it is negative or zero.

The reported Vth value is the current threshold voltage of the cell. The dvth value is the change in threshold voltage and the delvto value is the initial change in threshold voltage for the cell. The delvto value should correspond to the delvto value from the instance parameter.

## Examples

```
XA> iprint_flash_cell -dvth 0 -inst *
XF0.MCELL, Vth=-1.0986, dvth=-2.8487, delvto=0.5
XF1.MCELL, Vth=-0.74198, dvth=-2.4921, delvto=-0
XF2.MCELL, Vth=-0.92028, dvth=-2.6704, delvto=0.25
```

---

## iprint\_help

Displays the syntax and a brief description of the specified interactive commands.

## Syntax

```
iprint_help -cmd command_name1 ... command_namen
```

---

Argument	Description
<i>command_namen</i>	Specifies the interactive commands to be displayed on the screen.

---

## Examples

```
iprint_help iprint_node_info
```

This example prints information about the `iprint_node_info` command.

## **iprint\_node\_info**

Prints the node voltage, node index, and simulation time for the given node names. Each value is evaluated at the current simulation time, when the node voltage is last updated.

### **Syntax**

```
iprint_node_info -node node_name {node_name}
```

or

```
iprint_node_info -index node_index {node_index}
```

---

Argument	Description
-node node_name	Specifies node name. Supports wildcard characters.
-index node_index	Specifies node index.

---

### **Examples**

```
XA> iprint_node_info x1.sout
```

```
Node=X1.SOUT (225)
V=1.69417 V dV/dt=-0.00244323 V/ns t=1 ns
```

*Example 65*

```
iprint_node_info -node xalu3.xlatch2.q
```

[Example 65](#) prints information for node xalu3.xlatch2.q.

*Example 66*

```
iprint_node_info -index 1 3 2
```

[Example 66](#) prints node information for node indices 1, 3, and 2.

### **See Also**

[iprint\\_connectivity](#), [iprint\\_elem\\_info](#)

---

## **iprint\_subckt**

Prints the list of hierarchical instance names for all instances of the specified subcircuit. Note that this command does not support wildcard characters.

**Syntax**

```
iprint_subckt subckt_name [-file file_name]
```

Argument	Description
<i>subckt_name</i>	Specifies the subcircuit name.
-file <i>file_name</i>	Writes command output to the specified file.

**Examples**

```
XA> iprint_subckt nand2  
  
x1.x2.x3.xnand1  
x1.x2.x3.xnand2  
x1.x5.xnand1
```

---

**iprint\_time**

Prints the current simulation time.

**Syntax**

```
iprint_time
```

Argument	Description
N/A	

**Examples**

```
iprint_time
```

---

**iprint\_tree**

Prints information about the hierarchical instance tree. Only subcircuit instances are displayed. If the instance list is omitted, it is assumed to be an asterisk ( \* ). The output can also be dumped to a file.

**Syntax**

```
iprint_tree -inst inst_list  
[-limit val]  
[-a enable_value]
```

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

```
[-def enable_value]
[-file filename]
```

---

Argument	Description
-inst <i>inst_list</i>	Specifies an instance list. The underlying hierarchy of instances matching the pattern is printed.
-limit <i>val</i>	Specifies that the hierarchical depth to which the tree is printed. If not specified, this setting defaults to 3.
-a <i>enable_value</i>	Specifies that the full hierarchical name of each instance is displayed.
-def <i>enable_value</i>	Specifies that the subcircuit name is also displayed.
-file <i>filename</i>	Specifies that the command output is also written to the named file.

---

**Examples**

```
XA > iprint_tree -limit 0 -a 1 -def 1
x1 (dco_xtl)
x2 (dco)

XA > iprint_tree x1* -limit 1 -a 1 -def 1
x1 (main)
x1.xadd_204_u1 (muxi21x4)
x1.xadd_204_u7 (nor2x3)
```

---

**iprobe\_waveform\_current**

Creates device current waveform output.

**Syntax**

```
iprobe_waveform_current [[-i|i1] instance_name
                         {instance_name} [-in instance_name {instance_name}]
                         [-iall instance_name {instance_name}]
                         [-subckt subckt_name]
                         [-limit level]
                         [-delete enable_value]
```

---

<b>Argument</b>	<b>Description</b>
<code>-i i1 instance_name {instance_name}</code>	Specifies current through terminal 1. If you do not specify -i i1, the instance name must be the first argument. You can use wildcard characters in the instance name. Note that option applies only to instances not specified by -subckt.
<code>-in instance_name {instance_name}</code>	Specifies current through terminal <i>n</i> , where <i>n</i> is a positive integer. You can use wildcard characters in the instance name. Note that option applies only to instances not specified by -subckt.
<code>-iall instance_name {instance_name}</code>	Specifies current through all terminals. You can use wildcard characters in the instance name.
<code>-subckt subckt_name</code>	Specifies the subcircuit name of the specified instances to be probed.
<code>-limit level</code>	Specifies the hierarchy level. The default is 3.
<code>-delete enable_value</code>	If enabled, deletes the specified current probe instead of adding a current probe. the default off.

---

### Description

During interactive debugging, it is sometimes necessary to plot additional device currents for viewing that were not specified for printing in the netlist, and to remove other device currents. This command supports device current probing in interactive mode.

### Examples

In order to add/delete device current probes, you need to predefined the scope for possible device current probes with the `iprobe_waveform_current` command in the configuration file. Then, during the interactive mode, you can add the device current probes that you defined in the configuration file. You can delete the ones defined from the predefined scope and the ones defined in the netlist. For example, if you want to probe and delete some of the device currents under the `xd` sub-instance during the simulation, specify the following scope in the configuration file:

```
iprobe_waveform_current xd*
```

In interactive mode, you can probe any device current under that scope. For example:

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

```
iprobe_waveform_current xd.xcell0.*
```

Adds current probes of device matching `xd.xcell0.*`, such as `i(xd.xcell1.r1)` and `i(xd.xcell1.mn1)`.

```
iprobe_waveform_current -iall xd.xcell1.*
```

Adds current probes of devices matching `xd.xcell1.*`, such as `i1(xd.xcell1.r1)`, `i2(xd.xcell1.r1)`, `i1(xd.xcell1.mn1)`, `i2(xd.xcell1.mn1)`, `i3(xd.xcell1.mn1)`, and `i4(xd.xcell1.mn1)`.

**Note:** You need to set the scope using the `-iall` option in the configuration file.

```
iprobe_waveform_current xd.* -limit 2
```

Adds current probes of devices under `xd.*` below two hierarchical levels.

```
iprobe_waveform_current * -subckt subcell
```

Adds current probes of all devices under the `subcell` subcircuit.

You can also delete the specified current probe with:

```
iprobe_waveform_current xd.xcell0.* -delete 1
```

Deletes all device current probes under `xd.xcell0`.

```
iprobe_waveform_current xd.xcell0.m1i23  
-delete 1
```

Deletes the `xd.xcell0.m1i23` device current probe.

If you specify:

```
iprobe_waveform_current * -delete 1
```

This command deletes all the current probes. It includes all the interactive current probes (the probes added interactively) and the normal current probes.

---

**iprobe\_waveform\_voltage**

Creates a voltage waveform output.

## Syntax

```
iprobe_waveform_voltage -v node_name {node_name}
[-vn instance_name {instance_name}]
[-vall instance_name {instance_name}]
[-subckt subckt_name]
[-limit level]
[-port enable_value] [-index index {index}]
[-delete enable_value]
```

Argument	Description
-v node_name{ node_name}	Specifies the node name at which the voltage is probed.
-vn instance_name	Specifies the name of the instance at which the voltage of terminal <i>n</i> is probed, where <i>n</i> is a positive integer. Note this argument applies only to instances not specified by -subckt.
-vall instance_name	Specifies the name of the instance at which the voltage of all terminals are probed. Note this argument applies only to instances not specified by -subckt.
-subckt subckt_name	Probes the nodes in all instances of the named subcircuit. If you specify this argument, the nodes or instances should be in the subcircuit instance hierarchy level.
-limit level	Specifies the hierarchy level down to which the voltage is probed. When -subckt is specified, the -limit level is relative to where the particular node is located in the hierarchy. A value of 0 specifies the top level of the subcircuit. The default for level is 3.
-port enable_value	If you specify an enable_value value of 1, a wildcard character matches the subcircuit port names.
-delete enable_value	If enabled, removes the specified signals from the plot list.
-index index {index}	Writes the signals that match the specified indexes to the plot file.

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

**Description**

Probes the voltage on a node or on the pin of a primitive instance. The voltage waveform is written to the output file in the format specified by the `post` option in the netlist. Note that only the fsdb and wdf formats support adding new waveforms to the file on the fly.

You can use wildcards (\*) with the `-v`, `-vn`, and `-vall` arguments. When used with `-v`, the port alias matching is controlled by the `-port` argument or the [iset\\_interactive\\_option](#) setting. A `-port` specified with the command takes precedence. The [setWildcardRule](#) setting also applies.

Probes specified by this command are in addition to the `.probe` statement in the HSPICE or ELDO netlist files or `save` statements in the Spectre netlist files.

Long simulations, or simulations where some nodes have a high level of activity, can produce very large waveform files. To minimize waveform file loading time in these files, you can direct signals to separate waveform files and keep file sizes smaller.

If you use Custom WaveView to display the waveform file, you must close and reopen the file to be able to see the voltages you have added.

**Examples***Example 67*

```
XA> iprobe_waveform_voltage x1.*
```

[Example 67](#) adds probes for all nodes in instance `x1`. Wildcard matching is influenced by [setWildcardRule](#) and [iset\\_interactive\\_option](#).

*Example 68*

```
XA> iprobe_waveform_voltage * -delete 1
```

[Example 68](#) deletes all voltage probes.

*Example 69*

```
XA> iprobe_waveform_voltage x1.* -level 2
```

[Example 69](#) adds voltage probes for all nodes in instance `x1` down to level 2 of the hierarchy.

## iquit\_sim

Terminates the simulation.

### Syntax

iquit\_sim

Argument	Description
N/A	

### Examples

iquit\_sim

This example terminates the simulation.

## irelease\_node\_voltage

Releases the node voltages from the values fixed by [iforce\\_node\\_voltage](#). When you specify this command, the simulation results determine the node voltages.

**Note:** Nodes of voltage sources, Verilog-A outputs, and nodes that have been optimized out cannot be forced/released.

### Syntax

```
irelease_node_voltage -node node_name {node_name}
[-time time_val]
```

Argument	Description
-node node_name {node_name}	Releases voltages from the specified nodes.
-time time_val	Specifies the time when the nodes are released. The default is the current time.

### Examples

XA> irelease\_node\_voltage cn

The cn signal previously forced to a given value is released at the current time. The simulation results determine the cn voltage value until the end of the run

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

unless a new [iforce\\_node\\_voltage](#) command is specified.

---

**ireport\_node\_cap**

Reports capacitance information for the specified nodes.

**Syntax**

```
ireport_node_cap -node node_name {node_name}
    [-group group_name]
    [-limit limit_value]
    [-report basic|detail]
```

---

Argument	Description
-node node_name {node_name}	Reports capacitance for the node names you specify. You can use wildcard characters in the node names.
-group group_name	Creates a group name for the nodes you specify with -node. If you specify this option, all nodes for a report_node_cap command are grouped together. The CustomSim tool reports the capacitance information based on this group. Use this option only for a flat, postlayout design.
-limit limit_value	Specifies the hierarchy level down to which the CustomSim tool reports the capacitance information. The default is 3.
-report basic detail	Specifies the type of report to print. Use the basic keyword (the default) to print only the basic capacitance information. Use the detail keyword to print a detailed report.

---

**Description**

The reported node capacitance information includes total node capacitance, wire capacitance, gate capacitance of a MOSFET, and junction capacitance of a MOSFET. You can specify multiple commands in a simulation. The CustomSim tool processes each command separately.

report\_node\_cap outputs the capacitance information in a \* .cap# file.

In a prelayout design, capacitance is reported as:

$$C_{total} = C_{gate} + C_{junction} + C_{wire}$$

$$C_{gate} = C_{gd} + C_{gs} + C_{gb}$$

$$C_{junction} = C_{db} + C_{sb}$$

Cwire = Cdesign

In postlayout design, the CustomSim tool expands a single node from the prelayout design into multiple nodes because of the RC parasitics. The postlayout flow is divided into 2 scenarios:

- A back-annotated postlayout.
- A flat postlayout.

In the back-annotated postlayout flow, you can trace the connectivity back to the prelayout design with the information from the prelayout netlist and the back-annotation file:

```
* | NET na 0.00458507PF <-- Net Capacitance (CBAnet)
* | I (x02/mp:GATE x02/mp GATE I 4.8e-16 22.75 3.25) //
$llx=22.55 $lly=3.25 $urx=22.95 $ury=3.25 $lvl=5
* | I (x02/mn:GATE x02/mn GATE I 2.4e-16 22.75 1.05) //
$llx=22.55 $lly=0.6 $urx=22.95 $ury=1.05 $lvl=4
* | I (x01/mp:DRN x01/mp DRN B 0 8.45 3.25) // $llx=8.45
$lly=2.65 $urx=9.2 $ury=3.85 $lvl=7
* | I (x01/mn:DRN x01/mn DRN B 0 8.45 1.05) // $llx=8.45
$lly=0.75 $urx=9.2 $ury=1.35 $lvl=6
* | S (na:1 22.75 2.65) // $llx=22.55 $lly=2.65 $urx=22.95
$ury=2.65 $lvl=3
* | S (na:2 22.75 3.925) // $llx=22.55 $lly=3.85 $urx=22.95
$ury=4 $lvl=5
Cg1 na:1 0 4.33011e-17
Cg2 na:2 0 3.99892e-17
...
R158 na:1 x02/mp:GATE 4.8 $l=0.6 $w=0.4 $lvl=5
R159 na:1 na:3 30 $l=0.8 $w=0.4 $lvl=3
R160 x02/mp:GATE na:2 5.016 $l=0.675 $w=0.4 $lvl=5
R161 na:3 na:4 16.875 $l=0.5 $w=0.4 $lvl=3
R162 na:3 na:5 18.75 $l=0.5 $w=0.4 $lvl=3
R163 na:3 na:8 3.96 $a=0.04 $lvl=10
R164 na:4 na:6 3.96 $a=0.04 $lvl=1
```

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

```
...
R186 na:18 x01/mn:DRN 5.38888 $a=0.09 $lvl=12
R187 na:19 na:20 0.992002 $l=0.8 $w=0.5 $lvl=2
R188 na:19 na:21 13.1234 $l=6.45 $w=0.3 $lvl=2
R189 na:20 na:24 0.744002 $l=0.6 $w=0.5 $lvl=2
R190 na:20 x01/mp:DRN 5.38888 $a=0.09 $lvl=12
...
```

In back-annotated post-layout flow, the capacitance information is reported as:

$$C_{total} = C_{gate} + C_{junction} + C_{wire}$$

$$C_{gate} = C_{gd} + C_{gs} + C_{gb}$$

$$C_{junction} = C_{db} + C_{sb}$$

$$C_{wire} = CBA_{net} + C_{design}$$

Based on the previous back-annotation file example:

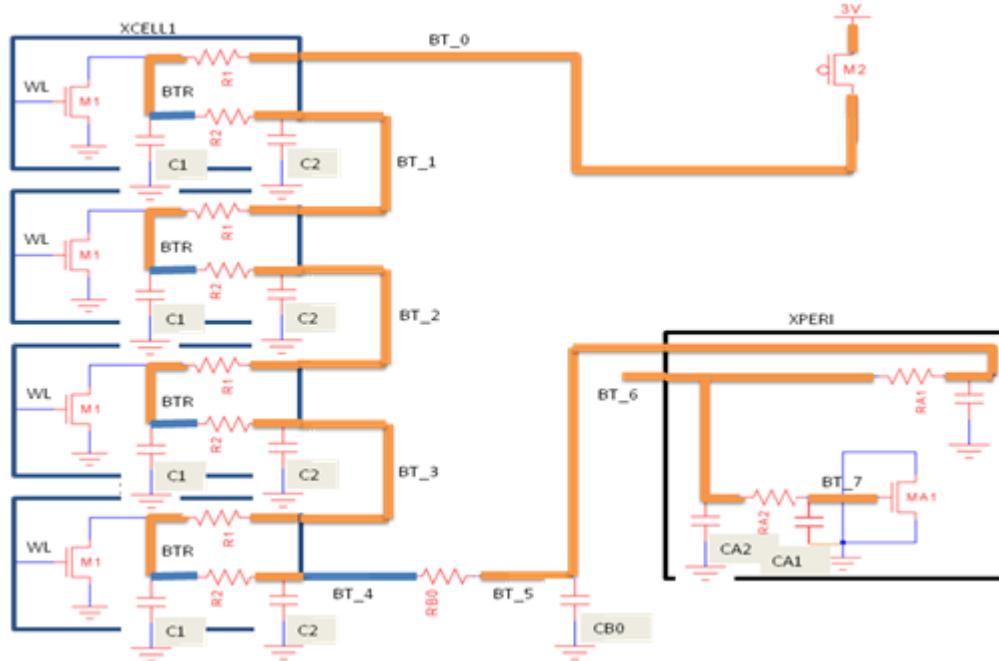
$$C_{total} = C_{gate} + C_{junction} + C_{wire}$$

$$C_{gate} = C_{gate}(x02/mp) + C_{gate}(x02/mn)$$

$$C_{junction} = C_{junction}(x01/mp) + C_{junction}(x01/mn)$$

$$C_{wire} = CBA_{net} (0.00458507PF) + C_{design}$$

In a flat postlayout flow, the CustomSim tool treats all nodes as unique and independent. To accurately calculate the capacitance information, you need to group the nodes and then run `report_node_cap`. For example, see [Figure 7](#).



*Figure 7 Flat Postlayout Flow Example*

In [Figure 7](#), a prelayout node, **BT**, has been expanded into different nodes in the postlayout. Because the postlayout netlist is flat, and there is no trace of connectivity from the prelayout netlist and back-annotation flow, all nodes are treated as unique and independent.

To accurately report the capacitance information, you need to tell the CustomSim tool which nodes can be grouped together for `report_node_cap` command to calculate the capacitance information. To group a list of nodes into one group, use `-group` argument and list the names of nodes to be grouped: **BT\_0**, **BT\_1**, **BT\_2**, **BT\_3**, **BT\_4**, **BT\_5**, **BT\_6**, **XPERI.BT\_7**, **XCELL1.BTR**, **XCELL2.BTR**, **XCELL3.BTR**, and **XCELL4.BTR** and do the following steps:

1. Specify the following `report_node_cap` command.

```
report_node_cap -node BT_? XPERI.BT_7 XCELL?.BTR -group BT
```

This command groups all the specified node names into one group named **BT**.

2. Assuming all parasitic capacitor has a value of 1 fF, the CustomSim tool calculates the capacitance information is as:

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

$$C_{total} = C_{gate} + C_{junction} + C_{wire}$$

$$C_{gate} = C_{gate}(XPERI.MA1)$$

$$C_{junction} = C_{junction}(M2) + C_{junction}(XCELL1.M1) + \\ C_{junction}(XCELL2.M1)$$

$$+ C_{junction}(XCELL3.M1) + C_{junction}(XCELL4.M1) + \\ C_{junction}(XPERI.MA1)$$

$$C_{wire} = (12 * 1 \text{ fF}) + C_{design}$$

$$C_{design} = 0 \text{ pF}$$

---

**ireport\_operating\_point**

Writes the circuit operating point at the current time to the specified file.

**Syntax**

```
ireport_operating_point -file filename
    [ -type ic | nodeset ]
    [-node node_name {node_name}]
```

---

Argument	Description
-file <i>file_name</i>	Specifies the report output file name.
-type ic   nodeset	Specifies if the file is a .ic or a .nodeset file. The default is .ic .
-node <i>node_name</i> { <i>node_name</i> }	Specifies that only those nodes matching the pattern are written to the file. You can specify wildcards in the node names. The wildcard match behavior is determined by the alias matching rules. See the <a href="#">iset_interactive_option -port</a> .description.

---

**Examples***Example 70*

```
XA> ireport_op op_100us.ic
```

The previous example writes the op\_100us.ic file with .ic statements for the default nodes available in the database.

*Example 71*

```
XA> ireport_op op_200us.nodeset -type nodeset
```

The previous example writes the `op_100us.nodeset` file with `.nodeset` statement for the default nodes available in the database.

*Example 72*

```
XA> ireport_op file1 -node add*
```

The previous example writes the `file1` file with `.ic` for those nodes matching the `add*` pattern from the default node set.

## **isearch\_node**

Searches nodes in the netlist and reports various attributes.

### Syntax

```
isearch_node -v [voltage_value] | -dv [dv_value] |
    -dt [dt_value] | -conn [conn_value]
```

Argument	Description
<code>-v voltage_value</code>	Specifies the maximum voltage. Any node with an absolute value higher than the specified <code>voltage_value</code> is reported. If you do not specify a value, the CustomSim tool reports the node with the highest voltage absolute value. The <code>voltage_value</code> unit is volts.
<code>-dv dv_value</code>	Specifies the maximum voltage change. Any node with a voltage change larger than the specified <code>dv_value</code> is reported. If you do not specify a value, the CustomSim tool reports the node with the highest voltage change. The <code>dv_value</code> unit is seconds.
<code>-dt dt_value</code>	Specifies the maximum time step. Any node with a time step smaller than the specified <code>dt_value</code> is reported. If you do not specify a value, the CustomSim tool reports the node with the smallest time step value. The <code>dt_value</code> unit is seconds.

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

Argument	Description
-conn <i>conn_value</i>	Returns the list of nodes that have more connections than the specified value. If you do not specify a value, it lists the most connected node in the circuit.

**Description**

This command searches the nodes in the netlist and reports nodes with:

- The highest voltage.
- All nodes with a voltage that exceeds the specified value.
- The highest voltage change.
- All nodes with a voltage change that exceeds the specified value.
- The nodes with the minimum time step.
- All nodes with a time step less than the specified value.
- The most connected node in the netlist.

`isearch_node` only counts connections to elements. A connection to a dangling subcircuit port does not count as a connection. In the following example, node *c* is connected only once to the `vc` voltage source.

```
.subckt dangling a
.ends
vc c 0 dc=1
xc c dangling
xc2 c dangling
```

The connected node output provides the name of the subcircuit that contains the node, the primary node name and node index, the number of connections, and a flag to indicate if a voltage source is connected to the node. For example:

```
XA > isearch_node -conn
Highest connectivity node: Subckt=con20 Node=a (1)
#Conn=21 Vsrc=0
```

```
XA > isearch_node -conn 10
```

```
High connectivity node: Subckt=TLC Node=0 (0) #Conn=14
Vsrc=1

High connectivity node: Subckt=TLC Node=a (1) #Conn=10
Vsrc=1

High connectivity node: Subckt=con20 Node=a (1) #Conn=21
Vsrc=0

High connectivity node: Subckt=con20 Node=c (3) #Conn=21
Vsrc=0
```

### **Examples**

```
isearch_node -v
Maximum V=4.567 sy nofr vcc_int
```

Prints the node with the maximum voltage.

```
XA> isearch_node -v 2.5
V=3 at node x8.qb
V=3 at node x8.CKn
V=3 at node vcc
V=3 at node d[2]
```

Reports all nodes with a voltage absolute value greater than 2.5V.

```
XA> isearch_node -dv
Maximum DV=0.276185 at node x2.x1.n2
```

Reports the node with the largest voltage change.

```
XA> isearch_node -dv 1e-3
DV=0.00243276 at node x3.x9.n1
DV=0.00230895 at node x3.x8.n1
```

Reports all nodes with a voltage change greater than 1mV.

```
XA> isearch_node -dt
Minimum time_step=33ps at node x4.x1.n2 , DV=0.272122
```

Reports the node with the minimum time step.

```
XA> isearch_node -dt 2000p
time_step=1934ps at node x3.x8.n1, DV=0.00230895
time_step=1934ps at node x1.x9.n1, DV=0.00144889
```

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

Reports the nodes with a time step smaller than 2000ps.

```
XA > isearch_node -conn  
Highest connectivity node: Subckt=con20 Node=a (1) #Conn=21 Vsrc=0
```

Reports the most connected node in the circuit.

```
XA > isearch_node -conn 10  
High connectivity node: Subckt=TLC Node=0 (0) #Conn=14 Vsrc=1  
High connectivity node: Subckt=TLC Node=a (1) #Conn=10 Vsrc=1  
High connectivity node: Subckt=con20 Node=a (1) #Conn=21 Vsrc=0  
High connectivity node: Subckt=con20 Node=c (3) #Conn=21 Vsrc=0
```

Reports all nodes that have 10 or more connections in the circuit.

---

**iset\_break\_point**

Pauses the simulation at the specified time.

**Syntax**

```
iset_break_point -at time[unit]
```

---

Argument	Description
<i>time[unit]</i>	Specifies the time and unit at which the simulation is paused. White space is not allowed between <i>time</i> and <i>unit</i> . The default for <i>unit</i> is second, if it is not specified.

---

**Examples**

```
iset_break_point -at 10n
```

This example pauses the simulation at 10ns.

---

**iset\_diagnostic\_option**

Lets you abort a simulation in progress and output the power report up to the current simulation time.

## Syntax

```
iset_diagnostic_option -report_power enable_value
```

---

Argument	Description
-report_power <i>enable_value</i>	This command only works with the <a href="#">report_power</a> command.

---

## Examples

```
report_power -label vdd_1u_2u -by_node vdd -from 1u -to 2u
xa net.sp -c cmd -intr 1u
XA> iprint_time
1us
XA> cont 500n
XA> iprint_time
1.5us
XA> iset_diagnostic_option -report_power 1
XA> iquit
```

The CustomSim tool completes the `vdd_1u_2u` power report using the `1.5us` as the window end time. the CustomSim tool needs to report the time value that was used to close the window for the calculation.

---

## iset\_env

Use this command when you want to see the entire result of your interactive command at once.

### Syntax

```
iset_env -filter|-filter output_filter
```

---

Argument	Description
-filter	Removes the page-by-page display of interactive results that requires using the space key.
-filter <i>output_filter</i>	Sets the output filter command, which can be filter for paging through the text, such as more, less, and so on. The default output filter command is <code>more</code> .

---

---

## iset\_interactive\_option

Controls the wildcard matching behavior in interactive mode for hierarchy and signal alias names. In interactive mode, this command overrides the wildcard-matching hierarchy set by the [setWildcardRule](#) batch mode command.

### Syntax

```
iset_interactive_option -match* one|all  
    [-port enable_value]  
    [-tcl enable_value]  
    [-tclbuf enable_value]
```

---

Argument	Description
-match* one all	Specifies whether the * wildcard character matches the hierarchical delimiter for one or all levels of hierarchy.
-port enable_value	Specifies whether the * wildcard character matches the signal alias names.
-tcl enable_value	Specifies whether the full Tcl interpreter is enabled. This option is the same as the -tcl command line argument. For more information about Tcl, see <a href="#">Enabling Tcl Mode</a> on page 3.
-tclbuf enable_value	Specifies if data from interactive mode commands is put in Tcl memory. Set -tclbuf 0 to avoid Tcl memory overflow in large circuits (default). To run Tcl scripts and enable data dumping, you need to set -tclbuf 1.

---

### Examples

```
XA> imatch_node x1.*  
7 X1.g  
8 X1.ha  
9 X1.x2.b1
```

This example matches 3 nodes with the x1.\* pattern.

```
XA> iset_interactive_option -match* one  
  
XA> imatch_node x1.*  
7 X1.g  
8 X1.ha
```

This example matches 2 nodes with the x1.\* pattern.

```
XA> iset_interactive_option -match* one -port 1  
  
XA> imatch_node x1.*  
  
0 X1.6  
0 X1.0  
1 X1.a  
1 X1.3  
2 X1.4  
3 X1.5  
7 X1.g  
8 X1.ha
```

This example matches 8 nodes with the x1.\* pattern.

---

## **iset\_interactive\_stop**

Enables the node voltage monitoring/checking capability.

### **Syntax**

```
iset_interactive_stop -check v(node_name) {v(node_name)}  
[-max vmax]  
[-min vmin]  
[-twindow tstart tstop {tstart tstop}]  
[-cmf script_file_name]
```

---

Argument	Description
-check v(node_name) {v(node_name)}	Specifies the voltage signals to be monitored during transient simulation.
-max vmax	If any monitored signal rises above the upper limit you specify, the CustomSim tool interrupts the transient simulation and enters interactive mode to facilitate debugging.

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

Argument	Description
<code>-min v<sub>min</sub></code>	If any monitored signal falls below the lower limit you specify, the CustomSim tool interrupts the transient simulation and enters interactive mode to facilitate debugging. Note that for monitored signals you should specify at least one <code>-max</code> or <code>-min</code> argument.
<code>-twindow t<sub>start</sub> t<sub>stop</sub></code> <code>{t<sub>start</sub> t<sub>stop</sub>}</code>	Specifies the time periods for the signals to be monitored in the simulation.
<code>-cmf script_file_name</code>	Specifies the script file to be run when the CustomSim tool enters interactive mode due to a violation at the upper or lower limits of <code>-max</code> or <code>-min</code> . Note that the commands in the script file must be interactive commands.

**Description**

If any monitored signal rises above the upper limit or falls below lower limit, the CustomSim tool interrupts the transient simulation and enters interactive mode. You can also use this command to specify a script file to run when entering interactive command due to either an upper/lower limit violation.

**Examples**

```
XA> iset_interactive_stop -check v(net1) -max 1.2 -cmf cmd_file  
XA> cont
```

Continues the transient simulation and stops when `v(net1)` is greater than 1.2V and applies the commands in `cmd_file`.

---

**iset\_save\_state**

Save a simulation at the specified time point.

**Note:** To use `iset_save_state` you must also set `set_save_state` in your command file.

**Syntax**

```
iset_save_state -time time_value
```

---

<b>Argument</b>	<b>Description</b>
-time <i>time_value</i>	Specifies the time point at which the state is saved to restore a simulation.

---

## **iset\_zstate\_option**

Sets the conducting rules for the [icheck\\_node\\_zstate](#) command.

### **Syntax**

```
iset_zstate_option [-rule rule_value {rule_value}]
[-diode_vth value]
[-idsth ids_value]
[-vbeth vb_value]
[-report report_value {report_value}]
```

---

<b>Argument</b>	<b>Description</b>
-rule <i>rule_value</i> { <i>rule_value</i> }	Selects the rule of conducting for a MOSFET. The MOSFET is considered conducting if one of the rules is met. Other rules are as follows:  NMOS: <ul style="list-style-type: none"><li>▪ <math>V_{gs} &gt; V_{th}</math> (rule=1)</li><li>▪ <math>I_{ds} &gt; idsth</math> (rule=2)</li><li>▪ <math>V_g &gt; V_{DD}-0.1</math> (rule=3)</li></ul> PMOS: <ul style="list-style-type: none"><li>▪ <math>V_{gs} &lt; V_{th}</math> (rule=1)</li><li>▪ <math>I_{ds} &gt; idsth</math> (rule=2)</li><li>▪ <math>V_g &lt; 0.1</math> (rule=3)</li></ul> The default is 1 and 2.
-diode_vth <i>value</i>	Specifies the threshold for a diode to be considered conducting when $V_{ac}$ exceeds the specified value. If you specify a value less than 0, diode is considered as not conducting. The default is 0.2V.
-idsth <i>ids_value</i>	Specifies the threshold for a MOSFET to be considered conducting when $ids$ exceeds <i>value_1</i> . If not specified, the default is 1e-8 ampere. See the Description section for more information about the conducting criteria.

**Chapter 3: CustomSim Interactive Command Syntax**

## CustomSim Interactive Commands

Argument	Description
<code>-vbeth vb_value</code>	Specifies the threshold for BJT to be considered conducting when Vbe exceeds <i>value_2</i> . If not specified, the default is 0.64 volt. See the Description section for more information about the conducting criteria.
<code>-report report_value {report_value}</code>	You can specify the following keywords: <ul style="list-style-type: none"> <li>▪ node lists high impedance nodes.</li> <li>▪ channel lists all channel-connected devices for the node.</li> <li>▪ gate lists all gate-connected devices for the node.</li> </ul> The default is <i>node</i> .

**Description**

This interactive command specifies the conducting rules so that the CustomSim tool can diagnose specified nodes staying in a high-impedance (floating) state.

A node stays in a high-impedance state if there is no conducting path from any voltage source to the node. A conducting path consists of conducting elements. An element is conducting if that specific element meets the following criteria:

Device	Rule
NMOS	<ul style="list-style-type: none"> <li>▪ <math>V_{gs} &gt; V_{th}</math> (-rule 1)</li> <li>▪ <math>I_{ds} &gt; idsth</math> (-rule 2)</li> <li>▪ <math>V_g &gt; VDD - 0.1</math> (-rule 3)</li> </ul>
PMOS	<ul style="list-style-type: none"> <li>▪ <math>V_{gs} &lt; V_{th}</math> (-rule 1)</li> <li>▪ <math>I_{ds} &gt; idsth</math> (-rule 2)</li> <li>▪ <math>V_g &lt; 0.1</math> (-rule 3)</li> </ul>
NPN	$V_{be} > vbeth$
PNP	$V_{be} < -vbeth$
Diode	Forward-biased.
Resistor	Always conducting.
Inductor	Always conducting.
Capacitor	Never conducting.

---

Device	Rule
Verilog-A	Never conducting.
Controlled sources	Always conducting.

---

## DC Interactive Mode Commands

DC interactive commands provide debugging tools to help you discover or identify issues in DC iterations. To enter DC interactive mode, use the `-dcintr` command line option. For example:

```
xa ... -dcintr iteration [-intr trans_time]
```

Where `iteration` specifies the number of DC iterations to run up to the specified iteration value and `trans_time` (optional) specifies the number of nanoseconds to run transient simulation, if desired.

The DC interactive mode commands are:

```
exit
iclose_log
icontinue_dc
idelete_node_ic
imatch_elem
imatch_node
iopen_log
iprint_connectivity
iprint_elem_info
iprint_exi
iprint_help
iprint_node_info
isearch_node
iset_node_ic
```

**Chapter 3: CustomSim Interactive Command Syntax**

## DC Interactive Mode Commands

You can also specify the HSIM DC interactive mode command syntax as shown in [Table 20](#). For information about the HSIM DC interactive mode commands, see the *HSIM Reliability Analysis Reference Manual*.

*Table 20 HSIM and CustomSim DC Interactive Commands*

HSIM Command	CustomSim Command
exit	<code>exit</code>
closelog	<code>iclose_log</code>
dccont	<code>icontinue_dc</code>
ric	<code>idelete_node_ic</code>
matche	<code>imatch_elem</code>
matchn	<code>imatch_node</code>
openlog	<code>iopen_log</code>
nc	<code>iprint_connectivity</code>
ev	<code>iprint_elem_info</code>
exi	<code>iprint_exi</code>
help	<code>iprint_help</code>
nv	<code>iprint_node_info</code>
dn	<code>isearch_node</code>
fic	<code>iset_node_ic</code>

---

**exit**

Stops and exits the simulation.

**Syntax**

`exit`

---

## **iclose\_log**

Closes the interactive mode log file that was opened by the [iopen\\_log](#) command. The log file contains all records of interactive mode commands and the results reported by the commands entered between `iopen_log` and `iclose_log`.

### **Syntax**

`iclose_log`

### **See Also**

[iopen\\_log](#)

---

## **icontinue\_dc**

Continues a DC simulation until it completes, stops at a specified iteration, or is interrupted by Ctrl-C.

### **Syntax**

`icontinue_dc [-iter addl_iteration] [-to iteration]  
[-to end]`

---

Argument	Description
<code>-iter addl_iteration</code>	Specifies the number of additional DC iterations to run based on the current DC iteration.
<code>-to iteration</code>	Runs the DC simulation until it reaches the specified iteration.
<code>-to end</code>	Runs the DC simulation until it reaches the DC converged point.

---

### **Examples**

XA DC> `icontinue_dc -iter 20`

Continues DC analysis for 20 additional iterations.

---

## **idelete\_node\_ic**

Releases initial condition values.

**Chapter 3: CustomSim Interactive Command Syntax**

## DC Interactive Mode Commands

**Syntax**

```
idelete_node_ic [-node] node_name
```

or

```
idelete_node_ic -index node_index
```

Argument	Description
-node <i>node_name</i>	Releases the initial condition value from the specified node. You can use a wildcard character in the node name.
-index <i>node_index</i>	Specifies the node index.

**Description**

This command releases the initial condition value from the .ic statement or [iset\\_node\\_ic](#) statement at DC interactive mode. This command does not work on voltage source nodes or vector files.

**Examples**

```
XA DC> idelete_node_ic -node n2 n5
```

---

**imatch\_elem**

Prints a list of the element indexes and hierarchical element names that match the specified pattern.

**Syntax**

```
imatch_elem -pattern pattern ...
```

Argument	Description
<i>pattern</i>	Patterns can contain the * wildcard character. The printed list shows all matches to the specified pattern. You can set wildcard matching for the hierarchy with <a href="#">iset_interactive_option</a> .

## Examples

```
XA DC> imatch_elem *x1*
5 x1.r1
6 x1.r2
7 x1.x1.r1
```

In this example, the `*x1*` pattern matches 3 elements.

## See Also

[iset\\_interactive\\_option](#)

---

## imatch\_node

Prints a list of the node indexes and node names that match the specified pattern.

### Syntax

```
imatch_node -pattern pattern ...
  [-limit level]
  [-port enable_value]
```

---

Argument	Description
<i>pattern</i>	Patterns can contain the <code>*</code> wildcard character. The printed list shows all matches to the specified pattern. You can set wildcard matching for the hierarchy with <a href="#">iset_interactive_option</a> .
<code>-limit <i>level</i></code>	Specifies wildcard matching only down to the specified hierarchical depth. The default hierarchical limit level is 3.
<code>-port <i>enable_value</i></code>	Enables or disables the matching of subcircuit port names. The default value of is taken from the <a href="#">iset_interactive_option</a> command.

---

### Description

`imatch_node` prints a list of matched nodes. The CustomSim tool first reports the value of `-limit` applied, then lists the node index and node name matched with one item per line. Finally, the CustomSim tool reports the total number of matched nodes.

**Chapter 3: CustomSim Interactive Command Syntax**

## DC Interactive Mode Commands

The [iset\\_interactive\\_option](#) command settings apply to this command. The "\*" wildcard can be set to match or not to match the hierarchical delimiter. The "\*" wildcard \* only matches primary node names, unless you specify the -port argument. When you specify -port, the "\*" wildcard also matches alias node names.

**Examples**

```
XA DC> imatch_node *a
1 a
2 x1.a
3 x1.fa
pattern *a matched 3 nodes
```

In this example, the \*a pattern matches 3 nodes.

```
XA DC> imatch_node *a -limit 0
imatch_node using -limit 0
1 a
pattern *a matched 1 nodes
```

This example finds all nodes ending in a at the top-level of the netlist, level 0.

```
XA DC> imatch_node *out -limit 4 -port 1
15 x0.x1.x2.aout
20 x0.x1.x2.x3a.out
35 x0.x1.x2.bout
48 x0.x1.x2.x3b.out
```

This example finds all nodes ending in out down to the hierarchical depth of 4 and also reports ports.

**See Also**

[iset\\_interactive\\_option](#)

---

**iopen\_log**

Opens the interactive mode *logfile\_name*, which contains the record of the interactive mode commands and the results reported by these commands until the log file is closed by the [iclose\\_log](#) command. Only one log file can be opened at one time.

**Syntax**

```
iopen_log -file logfile_name [-mode append|write]
```

Argument	Description
<code>-file logfile_name</code>	Specifies the interactive mode log file name.
<code>-mode append</code>	Specifies that contents are appended to pre-existing files using the same file name.
<code>-mode write</code>	Specifies that contents overwrite pre-existing files using the same file name. Default is <code>-mode write</code> .

**Examples**

XA DC> iopen\_log -file logfile

This example opens the log file named `logfile`.

**See Also**

[iclose\\_log](#)

---

**iprint\_connectivity**

Prints the detailed node connectivity information for the given node names or indices. The elements are categorized into channel-connected, gate-connected, and other elements.

**Syntax**

```
iprint_connectivity -node node_name {node_name}
  [-print gc|cc|o|all]
  [-on current_value]
  [-file file_name]
  [-file_append file_name]
```

or

```
iprint_connectivity -index node_index {node_index}
  [-print gc|cc|o|all]
  [-on current_value]
  [-file file_name]
  [-file_append file_name]
```

**Chapter 3: CustomSim Interactive Command Syntax**

## DC Interactive Mode Commands

Argument	Description
<code>-node node_name {node_name}</code>	Specifies the node name, which can contain wildcard characters.
<code>-print gc cc o all</code>	Specifies to selectively print the gate-connected, channel-connected, other, or all elements. If not specified, the default prints gate-connected and channel-connected elements.
<code>-index node_index</code>	Specifies node index.
<code>-on current_value</code>	Prints only those instances that have a current exceeding the specified <i>current_value</i> .
<code>-file file_name</code>	Writes output only to the specified file. If the file exists it is overwritten. The file is located in the directory specified by the <code>-o</code> command line argument.
<code>-file_append file_name</code>	Writes output only to the specified file. If the file exists this argument appends the output to the existing file, otherwise it creates the specified file. The file is located in the directory specified by the <code>-o</code> command line argument.

**Examples**

```
XA DC> iprint_connectivity -node xspine_0.xdqr_0.xl850.osc_2  
XA DC> iprint_connectivity xspine_0.xdqr_0.xl850.osc_2 -print cc  
XA DC> iprint_connectivity -index 3
```

**See Also**[iprint\\_node\\_info](#)

---

**iprint\_elem\_info**

Prints the detailed element information for the given element names at the specific time of activation.

## Syntax

```
iprint_elem_info -elem element_name {element_name}
  [-report brief]
  [-file file_name]
```

or

```
iprint_elem_info -index elem_index {elem_index}
  [-report brief]
  [-file file_name]
```

---

Argument	Description
-elem <i>element_name</i>	Specifies the element name, which can include wildcard characters.
-index <i>elem_indexn</i>	Specifies element index.
-report brief	Provides a brief version of the output that reports only the element terminal connectivity, voltage, and currents. Other information is suppressed.
-file <i>file_name</i>	Writes <code>iprint_elem_info</code> output to the specified file.

---

## Description

You can also provide a subcircuit instance as an element. In this case, the CustomSim tool prints the subcircuit name, the list of its ports, and the voltages on each port. The detailed element information includes:

- Element name, element type, and model name
- Element terminal connectivity
- Element parameters and values
- Element terminal voltages

MOSFET-specific information:

- MOS logic state (ON/OFF)
- MOS effective length and width ( $L_{eff}$  and  $W_{eff}$ )
- MOS conductance ( $g_{ds}$  and  $g_m$ )
- MOS threshold voltage ( $V_{th}$ )

**Chapter 3: CustomSim Interactive Command Syntax**

## DC Interactive Mode Commands

- MOS Voltage-dependent diode capacitance (`cbs` and `cbd`)
- MOS Voltage-dependent gate capacitance (`cgs`, `cgb`, and `cgd`)
- MOS  $I_{ds}$  current (`Ids`)

**Examples**

```
XA DC> iprint_elem_info x1.xi@1477.mpt1
```

```
Elem=X1.XI@1477.MPT1 (203) Type=PMOS Model=PENH  
D=X1.X001 (232) G=X1.N@1514 (130) S=VDD (14) B=VDD (14)
```

```
Vd=0.0108579 Vg=1.69226 Vs=1.69213 Vb=1.69213  
Weff=11.9178u Leff=0.205123u PD=13.4u PS=13.4u AD=4.2u^2 AS=4.2u^2  
Vt=-0.73418 OFF  
Ids=-2.7808e-07u  
gds=2.86161e-13 gm=6.111e-12  
cgs=3.03919f cgd=3.03903f cgb=4.59196f cbs=6.99465f cbd=4.48262f  
id=0.105008u ig=-0.0446823u is=0.000726083u ib=-0.0610515u
```

```
XA DC> iprint_elem_info X1.RI@5234
```

```
Elem=X1.RI@5234 (161) Type=R  
N1=X1.N@5446 (195) N2=X1.N@5445 (194)  
R=33  
Vn1=1.69267 Vn2=1.69267  
I=0.0036124u
```

*Example 73*

```
XA DC> iprint_elem_info -elem x1.x2.m1
```

[Example 73](#) prints information for element `x1.x2.m1`.

*Example 74*

```
XA DC> iprint_elem_info -index 1 3 2
```

[Example 74](#) prints information for element indices 1, 3, and 2.

*Example 75*

```
XA DC> iprint_elem_info x1  
Elem=x1 Type=subckt subckt=mysub1  
PORT 1 port=a - 11 (1) v=1  
PORT 2 port=b - 12 (2) v=1  
PORT 3 port=c - 13 (3) v=2.65  
PORT 4 port=d - 14 (4) v=1
```

[Example 75](#) prints subcircuit information and port voltages for a subcircuit instance.

### See Also

[iprint\\_connectivity](#), [iprint\\_node\\_info](#)

## iprint\_exi

Prints elements with excessive currents.

### Syntax

```
iprint_exi -inst inst_name {inst_name}
[-ith ivalue]
[-file file_name]
[-report report_value {report_value}]
```

Argument	Description
<code>-inst <i>inst_name</i></code>	Specifies the hierarchical element names to be searched. You can use wildcard characters (*) in the instance names. The <a href="#">set_wildcard_rule</a> conventions apply.
<code>-ith <i>ivalue</i></code>	Specifies the current threshold that must be exceeded for the element to be reported. The default value is 50uA.
<code>-file <i>file_name</i></code>	Specifies the output file name.
<code>-report <i>report_value</i> {<i>report_value</i>}</code>	<i>report_value</i> can be: <ul style="list-style-type: none"> <li>▪ subname to specify a list of subcircuit names of the hierarchical instance components to be printed in the output.</li> <li>▪ brief to report only element terminal connectivity, voltages, and currents. Other information is suppressed.</li> </ul>

### Description

`iprint_exi` reports the current through any device terminal that exceeds the threshold. The following device types are checked and reported:

- MOS
- Resistor

**Chapter 3: CustomSim Interactive Command Syntax**

## DC Interactive Mode Commands

- BJT
- Diode

**Examples**

The output format for elements found that exceed the current threshold is the same as [iprint\\_elem\\_info](#). If you specify a list of subcircuit names with the -report argument, the hierarchical element instance names have the following general format:

```
Elem=X0.X1.X2...Xn.modelname  
Subckt: X0=x0name X1=X1name X2=x2name ... Xn=Xnname
```

For example:

```
XA DC> iprint_exi -ith 1n xmos.m* -report subname  
Elem=Xmos.mn (6) Type=NMOS Model=nch.7  
D=vdd (5) G=g (3) S=Xmos.n1 (16) B=vb (4)  
Vd=3 Vg=1.81293e-90 Vs=-0.00837316 Vb=1.81293e-90  
M=1  
Weff=2.016u Leff=0.948223u PD=2.365u PS=2.365u AD=0.351792u^2  
AS=0.351792u^2 SA=0u SB=0u  
Vt=0.19898 OFF  
Ids=0.26352u  
gds=5.38557e-10 gm=0  
cgs=4.30157f cgd=0.416708f cgb=3.65854f cbs=1.20401f  
cbd=0.487088f  
id=0.26352u ig=-0.0663667u is=-0.00752308u ib=-0.189631u  
Subckt: Xmos=mymos
```

---

**iprint\_help**

Displays the syntax and a brief description of the specified interactive commands.

**Syntax**

```
iprint_help -cmd command_name1 ... command_namen
```

---

Argument	Description
<i>command_namen</i>	Specifies the interactive commands to be displayed on the screen.

---

**Examples**

```
XA DC> iprint_help iprint_node_info
```

This example prints information about the `iprint_node_info` command.

---

## iprint\_node\_info

Prints the node voltage, node index, iteration, and initial conditions (if any) for the given node names. Each value is evaluated at the current iteration, when the node voltage is last updated.

### Syntax

```
iprint_node_info -node node_name {node_name}
```

or

```
iprint_node_info -index node_index {node_index}
```

---

Argument	Description
-node <i>node_name</i>	Specifies node name. Supports wildcard characters.
-index <i>node_index</i>	Specifies node index.

---

### Examples

```
XA DC> iprint_node_info x1.sout
```

```
Node=X1.SOUT (225)
V=1.69417 V, dV=-0.00244323, iter=1, ic=0.5 V
```

#### *Example 76*

```
XA DC> iprint_node_info -node xalu3.xlatch2.q
```

[Example 76](#) prints information for node `xalu3.xlatch2.q`.

#### *Example 77*

```
XA DC> iprint_node_info -index 1 3 2
```

[Example 77](#) prints node information for node indices 1, 3, and 2.

### See Also

[iprint\\_connectivity](#), [iprint\\_elem\\_info](#)

## **isearch\_node**

Searches nodes in the netlist and reports various attributes.

### **Syntax**

```
isearch_node -v [voltage_value] | -dv [dv_value] |  
-conn [conn_value]  
-hiz enable_value
```

---

Argument	Description
-v <i>voltage_value</i>	Specifies the maximum voltage. Any node with an absolute value higher than the specified <i>voltage_value</i> is reported. If you do not specify a value, the CustomSim tool reports the node with the highest voltage absolute value. The <i>voltage_value</i> unit is volts.
-dv <i>dv_value</i>	Specifies the maximum voltage change. Any node with a voltage change larger than the specified <i>dv_value</i> is reported. If you do not specify a value, the CustomSim tool reports the node with the highest voltage change. The <i>dv_value</i> unit is seconds.
-conn <i>conn_value</i>	Returns the list of nodes that have more connections than the specified value. If you do not specify a value, it lists the most connected node in the circuit.
-hiz <i>enable_value</i>	When set to '0' prints all hiz nodes (default). When set to '1' prints all hiz nodes with fanouts.

---

### **Description**

This command searches the nodes in the netlist and reports nodes with:

- The highest voltage.
- All nodes with a voltage that exceeds the specified value.
- The highest voltage change.
- All nodes with a voltage change that exceeds the specified value.

- High impedance nodes.
- The most connected node in the netlist.

`isearch_node` only counts connections to elements. A connection to a dangling subcircuit port does not count as a connection. In the following example, node `c` is connected only once to the `vc` voltage source.

```
.subckt dangling a
.ends
vc c 0 dc=1
xc c dangling
xc2 c dangling
```

The connected node output provides the name of the subcircuit that contains the node, the primary node name and node index, the number of connections, and a flag to indicate if a voltage source is connected to the node. For example:

```
XA DC> isearch_node -conn
Highest connectivity node: Subckt=con20 Node=a (1)
#Conn=21 Vsrc=0

XA DC> isearch_node -conn 10
High connectivity node: Subckt=TLC Node=0 (0) #Conn=14
Vsrc=1
High connectivity node: Subckt=TLC Node=a (1) #Conn=10
Vsrc=1
High connectivity node: Subckt=con20 Node=a (1) #Conn=21
Vsrc=0
High connectivity node: Subckt=con20 Node=c (3) #Conn=21
Vsrc=0
```

### Examples

```
XA DC> isearch_node -v
Maximum V=4.567 sy nofr vcc_int
```

Prints the node with the maximum voltage.

**Chapter 3: CustomSim Interactive Command Syntax**

## DC Interactive Mode Commands

```
XA DC> isearch_node -v 2.5
V=3 at node x8.qb
V=3 at node x8.CKn
V=3 at node vcc
V=3 at node d[2]
```

Reports all nodes with a voltage absolute value greater than 2.5V.

```
XA DC> isearch_node -dv
Maximum DV=0.276185 at node x2.x1.n2
```

Reports the node with the largest voltage change.

```
XA DC> isearch_node -dv 1e-3
DV=0.00243276 at node x3.x9.n1
DV=0.00230895 at node x3.x8.n1
```

Reports all nodes with a voltage change greater than 1mV.

```
XA DC> isearch_node -conn
Highest connectivity node: Subckt=con20 Node=a (1) #Conn=21 Vsrc=0
```

Reports the most connected node in the circuit.

```
XA DC> isearch_node -conn 10
High connectivity node: Subckt=TLC Node=0 (0) #Conn=14 Vsrc=1
High connectivity node: Subckt=TLC Node=a (1) #Conn=10 Vsrc=1
High connectivity node: Subckt=con20 Node=a (1) #Conn=21 Vsrc=0
High connectivity node: Subckt=con20 Node=c (3) #Conn=21 Vsrc=0
```

Reports all nodes that have 10 or more connections in the circuit.

---

**iset\_node\_ic**

Changes initial condition values.

**Syntax**

```
iset_node_ic [-node] node_name -val value
```

or

```
-index node_index -val value
```

Argument	Description
-node <i>node_name</i>	Applies the initial conditions value to the specified nodes. You can use a wildcard character in the node name.
-val <i>value</i>	Specifies the initial conditions value.
-index <i>node_index</i>	Specifies the node index.

**Description**

This command changes initial condition value for the specified nodes. The specified nodes stay at the specified constant voltage. The node voltage stays at the same value from the current iteration until either DC convergence or when the constant node voltage status is released by the [idelete\\_node\\_ic](#) command. In the interactive mode, this command does not work on voltage source nodes or vector files.

**Examples**

```
XA DC> iset_node_ic -node pump -val 6.5
```

The voltage at the pump node stays at 6.5V from the current iteration on.

## **Feedback**

### **Chapter 3: CustomSim Interactive Command Syntax**

DC Interactive Mode Commands

## CustomSim Dynamic CircuitCheck

---

*Provides syntax for the CustomSim CircuitCheck (CCK) commands.*

The CCK commands are:

- [cck\\_excess\\_ipath](#)
- [cck\\_post](#)
- [cck\\_signal](#)
- [cck\\_soa](#)
- [cck\\_substrate](#)
- [cck\\_toggle\\_count](#)
- [cck\\_analog\\_pdown](#)
- [cck\\_resistive\\_path\\_report](#)

---

### cck\_excess\_ipath

Checks for excessive current paths from the specified starting nodes to the specified ending nodes. You can specify multiple `cck_excess_ipath` commands in a single simulation. Along the current path, CustomSim can support resistor, inductor, diode, MOSFET, and bipolar devices.

#### Syntax

```
cck_excess_ipath -label lname [scope] [-ith value]
[-ith2 value] [-tth value] [twindow] [from_node]
[to_node] [-at time {time}] [-file merge|split]

scope ::= [inst] [except_i]
```

**Chapter 4: CustomSim Dynamic CircuitCheck****cck\_excess\_ipath**

```
inst ::= -inst inst_pattern {inst_pattern}  
except_i ::= -except_inst inst_pattern {inst_pattern}  
twindow ::= [-twindow {tstart tstop} tstart [tstop]  
[-tstep tstep_val] [-tdelay tdelay_val]  
from_node ::= -from_node node_pattern {node_pattern}  
to_node ::= -to_node node_pattern {node_pattern}
```

Argument	Description
-label label_name	Specifies the label name that appears in the report file, which makes it easier to search the report.
-inst inst_pattern {inst_pattern}	Scopes to the named instances. You can use wildcard characters in the instance names according to the rules specified by the set_wildcard_rule command.
-except_inst inst_pattern {inst_pattern}	Excludes instances from the command scope. Any instance whose hierarchical instance name matches the pattern is excluded. You can use wildcard characters in the instance names according to the rules specified by the set_wildcard_rule command.
-ith value	Specifies the current threshold of an excessive current path. The elements in a path must have a current value that exceeds the specified value. For a violation to be reported, this option takes absolute value. The default is 50 uA or 5.0e-5.
-ith2 value	Specifies the threshold current for an additional remark on the DC path generated by -ith. If MOSFET devices are on the DC path and if their Ids current exceeds the specified (or default) -ith2 value, the command will print an additional remark under the device Status section ( $I > ith2$ ). The default ith2 value is 1e-3A.
-tth value	Specifies the time duration of an excessive current path. For a violation to be reported, the current in an excessive current path must last longer than the specified value. The default is 5 ns or 5.0e-9.

Argument	Description
-twindow { <i>tstart tstop</i> } <i>tstart tstop</i> -tstep <i>tstep_val</i>	Instructs CustomSim to perform the check within the time window defined by <i>tstart tstop</i> . The <i>tstart</i> and <i>tstop</i> must come in pairs, except for the final window where: if <i>tstop</i> is not specified, it is assumed to be the end of the simulation. The final <i>tstop</i> can also be the end or END keyword.
-tstep <i>tstep_val</i>	If this argument is specified, the cck_excess_ipath check is performed at discrete time points, which are integer multiples of <i>tstep_val</i> . You can use this argument in conjunction with the -twindow argument. When you specify this argument, the -tth argument is ignored.
-tdelay <i>tdelay_val</i>	Performs checking at each time defined as <i>t+tdelay_val</i> . "t" is the time an input voltage source changes to a new voltage level. You can use this option with -twindow.
-from_node <i>node_pattern</i> { <i>node_pattern</i> }	Specifies the starting nodes of a path to be checked, which avoids long simulation times. Wildcard characters in the node names are supported for nodes that have a direct connection to the voltage source.
-to_node <i>node_pattern</i> { <i>node_pattern</i> }	Specifies the ending nodes of a path to be checked, which avoids long simulation times. Wildcard characters in the node names are supported for nodes that have a direct connection to the voltage source.
-at <i>time</i> { <i>time</i> }	Performs discrete checking at the specified time points. When you specify this argument, the output from the measurement is redirected to another file (.cckexipath_label) where <i>label</i> is the label name of the statement.
-file merge split	Enables file merging or splitting. When set to <i>merge</i> (the default), all outputs are merged into one output file (named <i>prefix.cckexipath</i> ). When set to <i>split</i> , all the outputs are split into different files (named <i>prefix.cckexipath_label</i> ).

**Chapter 4: CustomSim Dynamic CircuitCheck**

cck\_post

**Description**

If you do not specify `-from_node` and `-to_node`, `cck_excess_ipath` searches the DC paths with all combinations of all voltage sources (VSRCs). To avoid a long simulation time, specify the starting nodes and ending nodes.

At the end of the simulation, violations are reported to an output file,  
`*.cckexipath`.

**Examples***Example 78*

```
cck_excess_ipath -label test1 -ith 10u -tth 0.6n -from_node vdd
xam.d1 -to_node gnd -twindow 10n 100n
```

Monitors the paths from `vdd` or `xam.d1` to `gnd` between 10 –100 ns. If all the elements in these paths have currents greater than 10 uA, and last for more than 0.6 ns, a violation is reported.

**Note:** Due to the embedded CustomSim RC optimization techniques, some resistors in the RC network are trimmed. This impacts the quality of the report generated by the dynamic CCK path analysis commands, such as `cck_analog_pdown` and `cck_excess_ipath` as the expected resistors along the reported path are unavailable.

The command, `cck_resistive_path_report` can be applied together with the dynamic CCK path analysis commands to retain the original resistors and to ensure quality of the report. For more information see, [cck\\_resistive\\_path\\_report on page 354](#)

---

**cck\_post**

Post processes results from the `cck_soa` and the `cck_signal` commands. `cck_soa` and `cck_signal` commands support voltage and current checking.

**Syntax**

```
cck_post -switch enable_value [-waveform file_name]
enable_value ::= 1 | 0 | on | off | yes | no | true | false
```

Argument	Description
<code>enable_value</code>	Enables/disables postprocessing.
<code>-waveform file_name</code>	Reads the waveform file and runs postprocessing. If the waveform file does not exist, CustomSim issues an error message. Note that split waveforms are not supported for <code>cck_soa</code> and the <code>cck_signal</code> postprocessing.

### Description

The `cck_post` command runs all the diagnostic circuit checks performed at the end of the simulation based on a waveform file. It supports the postprocessing results from the `cck_soa` and the `cck_signal` commands.

**Note:** The `cck_post` command supports on the FSDB and WDF output formats.

**Chapter 4: CustomSim Dynamic CircuitCheck**

cck\_post

CustomSim supports postprocessing CCK results using a two-pass or single-pass method:

- In the two-pass method, the first pass dumps the signals into the waveform file using the probing commands for voltage and current checking, `probe_waveform_voltage` and `probe_waveform_current`. For example:

```
probe_waveform_voltage -vall * -limit 100  
probe_waveform_current -iall * -limit 100
```

If you know the specific signals that you want to use for postprocessing, you can limit the probing to only those signals. During postprocessing, if a signal called by the `cck_soa` or the `cck_signal` command is not found, a warning is issued.

- In the single-pass method, CustomSim automatically dumps only the necessary signals in the waveform file based on the scope specified by the `cck_soa` and the `cck_signal` commands. This method generates a smaller waveform file. For example:

```
cck_post ...  
cck_soa -label test1 -check vgs ...  
cck_soa -label test2 -check id ...  
cck_signal -label test1 -check vgs ...  
cck_signal -label test2 -check i1 ...
```

## Examples

### *Example 79 Single-Pass Example*

```
cck_post -switch 1
```

### *Example 80 Second-Pass Example*

```
cck_post -switch 1 -waveform test.fsdb
```

The previous example enables the second-pass of postprocessing by reading in the `test.fsdb` waveform file.

---

## cck\_signal

Checks the state of an arbitrary collection of signals that can be part of the checks for node and instance probes. The `cck_signal` command also supports current signals.

### Syntax

```
cck_signal -label lname [scope] [-numv num_val]
    constraint [twindow] [-flush auto | auto2]
    [-limit level] [-filterAlert 0/1] [-portalert
    enable_value] [-numvd num_value] [-report conn|subinfo]

scope ::= [inst] [subckt] [except_i] [except_s]

inst ::= -inst inst_pattern {inst_pattern}

subckt ::= -subckt subckt_name {subckt_name}

except_i ::= -except_inst inst_pattern {inst_pattern}

except_s ::= -except_subckt subckt_name[.inst_pattern]
    {subckt_name[.inst_pattern]}

constraint ::= real_value_check|logic_check

real_value_check ::= -check real_expression -min min_value
    -max max_value [-tth value]

logic_check ::= -check logic_expression [-logic safe |
    violation] [-tth value]

twindow ::= [-twindow {tstart tstop} tstart [tstop]]
    [-tstep tstep_val]

tstop ::= time_value|end|END

report ::= -report report_type

report_type ::= operand
```

## Chapter 4: CustomSim Dynamic CircuitCheck

cck\_signal

Argument	Description
<code>-label <i>label_name</i></code>	Specifies the label name that appears in the report file, which makes it easier to search the report. If on-the-fly reporting is enabled, the label forms part of the report filename.
<code>-inst <i>inst_pattern</i> {<i>inst_pattern</i>}</code>	Scopes to the named instances. You can use wildcards in instance names as specified by the CustomSim <code>set_wildcard_rule</code> command. Wildcard behavior is determined by the <code>set_wildcard_rule</code> command.
<code>-subckt <i>subckt_name</i> {<i>subckt_name</i>}</code>	Scopes to instances of the named subcircuits.
<code>-except_inst <i>inst_pattern</i> {<i>inst_pattern</i>}</code>	Excludes instances from the <code>cck_signal</code> command scope. Any instance whose hierarchical instance name matches the pattern is excluded from the scope. Wildcards are supported.
<code>-except_subckt <i>sub_name</i>[.<i>inst_pattern</i>] {<i>sub_name</i>} [.<i>inst_pattern</i>]</code>	Excludes all instances of the named subcircuits from the <code>cck_signal</code> command scope. Any subcircuit whose name matches the pattern is excluded from the scope. You can also exclude only the named instances of the subcircuit. Wildcards are not supported in the subcircuit names, but are supported in the <i>inst_pattern</i> .  Use the following syntax to exclude multiple subcircuits in the same <code>cck_signal</code> command statement: <code>cck_signal...-except_subckt sub_a sub_b sub_c</code> Where <code>sub_a</code> , <code>sub_c</code> , and <code>sub_c</code> are the explicit subcircuit names to be excluded.
<code>-numv <i>num_val</i></code>	Limits the maximum number of total violations reported. The default is 300. To print all violations, use the <code>all</code> keyword to print all violations. To turn off violation reporting, specify <code>0</code> .

Argument	Description
<code>-check real_expression -min min_value -max max_value [-tth tval]   logic_expression [-logic safe   violation] [-tth tval]</code>	You must provide a constraint expression, which can be a logic expression or a real-value expression (a real-value expression can also be an arithmetic expression). If the expression contains a logical operator, it is considered to be a logic expression.  Any logic expression that evaluates to logical false (0) for a period longer than the <code>tval</code> , is reported as a violation. The logical expression uses the perspective of the defining safe operating area. You can change it with the <code>-logic</code> option. The default for <code>-logic</code> is set to <code>safe</code> . For information about the <code>safe</code> and <code>violation</code> keywords, see <a href="#">Table 21</a> .
	A constraint that is a real-value expression is reported as a violation if it is outside the safe operating range specified by <code>-min</code> and <code>-max</code> arguments.
	To conform to Tcl conventions, enclose the constraint expression in double quotes or braces if it contains any white space.
	This expression is defined as a single group for the Tcl parser. This means that it should not contain white space or be grouped with the Tcl grouping operators, double quotes (""), or braces ({}).
	You can construct the expression with a single or a group of constraint functions (see <a href="#">Table 22</a> ) with either arithmetic operators (+, -, *, /) or logical operators (>, <, >=, <=, &&,   , ==, !=).
<code>-twindow {tstart tstop} tstart tstop -tstep tstep_val</code>	If you specify this argument, CustomSim performs the <code>cck_signal</code> check within the time window defined by <code>tstart tstop {tstart tstop}</code> . The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where if <code>tstop</code> is not specified, it is assumed to be the end of the simulation. The final <code>tstop</code> can also be the <code>end</code> or <code>END</code> keyword.  The default is from <code>tstart=0</code> to <code>tstop=end</code> .

**Chapter 4: CustomSim Dynamic CircuitCheck**

## cck\_signal

Argument	Description
<code>-tstep <i>tstep_val</i></code>	If this argument is specified, the <code>cck_signal</code> check is performed at discrete time points, which are integer multiples of <code><i>tstep_val</i></code> . You can use this argument with the <code>-twindow</code> argument. When you specify this argument, the <code>-tth</code> argument is ignored.
<code>-flush auto   auto2</code>	Auto-flushing is enabled by providing the <code>auto2</code> keyword, which specifies to report violations according to the following criteria: <ul style="list-style-type: none"><li>▪ Report violations as they occur.</li><li>▪ Report violations if the violation duration satisfies the time period specified by the <code>cck_signal</code> command constraint expression.</li></ul>
<code>-filterAlert 0/1</code>	Specifies to issue the <code>zero scoping error</code> message if there are no devices that match with the user specified scoping rule. This can be caused by an unintended typo or the matched device might be excluded by the <code>-except_inst</code> or the <code>-except_subckt</code> expression in the same <code>cck_signal</code> statement. By default, ( <code>-filterAlert 1</code> ) the command considers such a scenario as an error and therefore displays the <code>zero scoping error</code> message and terminates the simulation. If you specify <code>-filterAlert 0</code> , only the <code>zero scoping error</code> message is displayed and the simulation is not terminated.
<code>-limit <i>level</i></code>	Specifies the hierarchy level to which the scope is applied. When you specify <code>-subckt</code> , the <code>-limit</code> is relative to where the subcircuit is located in the hierarchy. A value of 0 specifies the top level of the subcircuit. The default is for all levels. This option is effective only when scoping includes wildcard characters.

Argument	Description
<code>-portalert enable_value</code>	<p>Displays a warning or error message if the Verilog-A port specified in the constraint expression is not matched.</p> <ul style="list-style-type: none"> <li>▪ If you specify 1, an error message is displayed and the simulation is terminated.</li> <li>▪ If you specify 0, a warning message is displayed and the simulation is continued.</li> </ul> <p>The default is 1.</p> <p>You can set <code>enable_value</code> as 1   0, yes/no or true/false.</p>
<code>-report report_type</code>	<p>Enables the following additional information to be reported for each device.</p> <ul style="list-style-type: none"> <li>▪ <code>conn</code> specifies to report connectivity information.</li> <li>▪ <code>subinfo</code> specifies to report the subcircuit hierarchy information.</li> </ul>
<code>-numvd num_val</code>	<p>Limits the number of violations per device or node that are reported. The default is 1. To print all violations, specify the <code>all</code> keyword. To turn off violation reporting, specify 0.</p> <p>If the check expression is device based, that is, <code>-check {v(c1) - v(c2) &lt;= 0.2}</code>, then the <code>-numvd</code> option specifies the number of violations per device.</p> <p>If the check expression is node based, that is <code>-check {v(node1) &lt;= 0}</code>, then the <code>-numvd</code> option specifies the number of violation per node.</p>

### Description

You can use the "\*" wildcard character in the check expression in a voltage access function if the other terms of the expression have a constant value. The following scenarios are supported:

```
v(*)
"v(*) - 3.0"
"v(top.x1.*.port.*)/2"
```

The following scenarios are not supported because they contain a second signal access function:

```
"v(*) - v(node1) + 1.2"
"v(*) - v(node1) + 1.2"
```

## Feedback

### Chapter 4: CustomSim Dynamic CircuitCheck

cck\_signal

**Table 21** lists the conditions for the `-logic safe` and `violation` keywords.

*Table 21 -logic conditions*

Expression Value/Logic Setting	<code>-logic safe (default)</code>	<code>-logic violation</code>
TRUE		Report a violation.
FALSE		Report a violation.

**Table 22** lists the accessing common instance conditions.

*Table 22 Constraint Functions*

Constraint Function	Description
<code>abs(x)</code>	Returns the absolute value of <code>x</code> : $ x $ .
<code>ceil(x)</code>	Returns the integer that is greater than or equal to value of <code>x</code> .
<code>Vg(inst)</code>	Gate voltage
<code>Vd(inst)</code>	Drain voltage
<code>Vs(inst)</code>	Source voltage
<code>Vb(inst)</code>	Bulk voltage
<code>Vgs(inst)</code>	Gate-source voltage
<code>Vsg(inst)</code>	Source-gate voltage
<code>Vds(inst)</code>	Drain-source voltage
<code>Vsd(inst)</code>	Source-drain voltage
<code>Vgd(inst)</code>	Gate-drain voltage
<code>Vdg(inst)</code>	Drain-gate voltage
<code>Vdb(inst)</code>	Drain-bulk voltage
<code>Vbd(inst)</code>	Bulk-drain voltage
<code>Vsb(inst)</code>	Source-bulk voltage

*Table 22 Constraint Functions*

<b>Constraint Function</b>	<b>Description</b>
Vbs(inst)	Bulk-source voltage
iN(inst)	N-terminal current
V(node)	Node voltage
X(inst.port)	Subcircuit port current
Isub(inst.port)	Subcircuit port current

**Table 23** lists the mathematical functions are supported to construct the constraint expressions.

*Table 23 Mathematical Functions*

<b>Built-In Function</b>	<b>Description</b>
abs(x)	Returns the absolute value of x: $ x $ .
ceil(x)	Returns the integer that is greater than or equal to value of x.
exp(x)	Returns e, raised to the power of the value of x: $e^x$ .
floor(x)	Returns the integer that is less than or equal to value of x.
log(x)	Returns the natural logarithm of the absolute value of x, with the sign of x: $(\text{sign of } x)\log( x )$ .
log10(x)	Returns the base 10 logarithm of the absolute value of x, with the sign of x: $(\text{sign of } x)\log_{10}( x )$ .
pow(x,y)	Returns the value of x raised to the power of the value of y.
sqrt(x)	Returns the square root of the absolute value of x: $\sqrt{-x} = -\sqrt{ x }$ .

To conform to the Tcl convention, enclose the constraint expression in double quotes or braces if it contains any white space.

**Chapter 4: CustomSim Dynamic CircuitCheck****cck\_signal**

You can use `cck_signal` to scope to specific instances using the `-inst`, `-subckt`, `-except_inst`, and `-expect_subckt` arguments. You can use wildcards only with the `-inst` and `-except_inst` arguments. Wildcard characters are not supported for the `-subckt` and `-except_subckt` arguments.

The rule between `-except_subckt` and `-except_inst` arguments accumulates in a logical OR. If you use the `-except_subckt` argument with `-except_inst`, the checks exclude all the instances inside the named subcircuit by the `-except_subckt` argument and all named instances by `-except_inst`.

When you specify the `-flush` argument, on-the-fly auto-reporting is enabled. Each `cck_signal` command for which auto-flushing is enabled writes its output to a unique file. These files are named as `prefix.ccksoa.label`, where `prefix` is the input netlist filename `prefix`, or the name specified with `-o` on the command line. The label is the command label specified by `-label` argument of that command. Each file is updated as violations are detected.

**Examples***Example 81 Basic Check*

```
cck_signal -label simple_check -check vgs(x1.m1) -min 0 -max 0.6
```

*Example 82 Logic Check*

```
cck_signal -label logic_check -check {VGS(x1.m1) < 0.3 &&  
v(x1.vddhp) > 3.0}
```

Checks a MOS terminal voltage and a node voltage.

*Example 83 Timing Check*

```
cck_signal -label simple_check_3n -subckt regulator -check  
v(vreg) -min 1.1 -max 1.2 -tth 10n
```

Flags a violation if `v(vreg)` is outside the `-min/-max` values for more than 10 ns.

*Example 84 Timing Window Check*

```
cck_signal -label window_check -subckt regulator -check v(creg)  
-min 1.1 -max 1.2 -twindow 1u 2u 3u 4u
```

Checks two windows: the first is from 1– 2 us, and the second is from 3 – 4 us.

*Example 85 Discrete Points Check*

```
cck_signal -label step_check -check v(x1.vreg) -min 0 -max 0.6 -
tstep 500n
```

*Example 86 Timing Window Check with Step*

```
cck_signal -label window_step -inst x1 -check v(ctrl) -min 0 -
max 0.6 -twindow 500n 5u -tstep 200n
```

*Example 87 Except Instance Pattern Check*

```
cck_signal -label except_inst_check -subckt mysub -except_inst
x1.xsub1 x1.xsub2 -check v(ctrl) -min 0.5 -max 0.75
```

Checks v(ctrl) on every instance of mysub, except x1.xsub1 and x1.xsub2.

*Example 88 Except Subcircuit Pattern Check*

```
cck_signal -label except_subckt_check -inst xctrl_top.xcmod* -
except_subckt dummy_ctrl -check v(vout) -min 0.5 -max 1.2
    xctrl_top a b c d .... ctrl_top
    .subckt ctrl_top

    xcmod0 1 2 3 log_ctrl
    xcmod1 4 5 6 log_ctrl
    xcmod2 7 8 9 log_ctrl
    xcmod3 10 11 12 dummy_ctrl
    xcmod4 ... dummy_ctrl
    ...
    xcmod15 ... dummy_ctrl
    .ends
    .subckt log_ctrl a b c
    ro vout c r=10
    ...
    .ends
    .subckt dummy_ctrl a b c
    ro vout c r=10
    evout vout 0 vcv a b 0.5
    .ends
```

In this example, the `-inst` option scopes to xcmod0, xcmod1, xcmod2, ... xcmod15. The `-except_subckt` option removes xcmod4 through xcmod15. The resulting checked signals are:

- `xctrl_top.xcmod0.vout`
- `xctrl_top.xcmod1.vout`

**Chapter 4: CustomSim Dynamic CircuitCheck**

cck\_soa

- `xctrl_top.xcmod2.vout`
- `xctrl_top.xcmod3.vout`

**Example 89 Error on Zero Scope Condition Check**

```
cck_signal -inst xctrl_top.xcmod* -check v(vout) -min 0.5 -max 1.2

xctrl_top a b c d ... ctrl_top
  .subckt ctrl_top
    xcmod0 1 2 3 log_ctrl
    xcmod4 .... dummy_ctrl
    .ends
    .subckt log_ctrl a b c
    ro vout c r=10
    ...
    .ends
    .subckt dummy_ctrl a b c
    *does not contain any noted vout
    ro cout c r=10
    evout cout 0 vcv a b 0.5
    .ends
```

In this example, the wildcard in `-inst` scopes to `xcmod0` and `xcmod4`, but the `vout` signal does not exist in `xcmod4`, so CustomSim raises a zero scope error.

---

**cck\_soa**

Runs a safe operating area check.

**Syntax**

```
cck_soa -label lname [scope] [-numvd num_val]
  constraint
  [twindow] [-flush auto|auto2] [report]
  [-numd num_val] [-filteralert 0/1]
  [-portalert enable_value] [-limit value]

  scope ::= [model] [inst] [subckt] [instparam] [except_i]
          [except_s]

  model ::= -model model_name {model_name}

  inst ::= -inst inst_pattern {inst_pattern}
```

```

subckt ::= -subckt subckt_name {subckt_name}

instparam ::= -instparam inst_expression

except_i ::= -except_inst inst_pattern {inst_pattern}

except_s ::= -except_subckt subckt_name[.inst_pattern]
               {subckt_name[.inst_pattern]}

constraint ::= real_value_check|logic_check

real_value_check ::= -check real_expression -min min_value
                         -max max_value [-tth value]

logic_check ::= -check logic_expression [-logic safe |
                                         violation] [-tth value]

twindow ::= [-twindow {tstart tstop} tstart [tstop]]
                  [-tstep tstep_val]

report ::= -report report_type {report_type}

report_type ::= wl | conn | subinfo | operand

enable_value ::= 1 | 0 | on | off | yes | no | true | false

```

---

Argument	Description
<b>-label</b> <i>label_name</i>	Specifies the label name that appears in the report file, which makes it easier to search the report. If on-the-fly reporting is enabled, the label forms part of the report filename.
<b>-model</b> <i>model_name</i> { <i>model_name</i> }	Scopes to devices of the named models. <b>all_resistors</b> , <b>all_capacitors</b> , and <b>all_diodes</b> are reserved keywords that are used to scope to all the resistors, capacitors, and diodes respectively. Wildcard expression is supported.

## Chapter 4: CustomSim Dynamic CircuitCheck

cck\_soa

Argument	Description
<code>-inst <i>inst_pattern</i> {<i>inst_pattern</i>}</code>	Scopes to the named instances. You can use wildcards in instance names as specified by the CustomSim <code>set_wildcard_rule</code> command. Wildcard behavior is determined by the <code>set_wildcard_rule</code> command.
<code>-subckt <i>subckt_name</i> {<i>subckt_name</i>}</code>	Scopes to instances of the named subcircuits. When you use the <code>-subckt</code> argument, the <code>-model</code> and <code>-inst</code> arguments become local to the named subcircuits.
<code>-instparam <i>inst_expression</i></code>	As an expression of w and l, this argument further scopes to devices whose length and width cause the expression to evaluate to logical true (1). Note that this argument only provides additional scoping to the <code>cck_soa</code> command. To conform to Tcl conventions, its expression should be enclosed within double quotes or braces if it contains any white space.  This expression is defined as a single group for the Tcl parser. This means that it should not contain white space or be grouped with the Tcl grouping operators, double quotes (""), or braces ({}).
<code>-except_inst <i>inst_pattern</i> {<i>inst_pattern</i>}</code>	Excludes instances from the <code>cck_soa</code> command scope. Any instance whose hierarchical instance name matches the pattern is excluded from the scope. Wildcards are supported.
<code>-except_subckt <i>sub_name</i>[.<i>inst_pattern</i>] {<i>sub_name</i>}[.<i>inst_pattern</i>]</code>	Excludes all instances of the named subcircuits from the <code>cck_soa</code> command scope. Any subcircuit whose name matches the pattern is excluded from the scope. You can also exclude only the named instances of the subcircuit. Wildcards are not supported in subcircuit names, but are supported in the <i>inst_pattern</i> .  Use the following syntax to exclude multiple subcircuits in the same <code>cck_soa</code> command statement: <code>cck_soa...-except_subckt sub_a sub_b sub_c</code> Where <code>sub_a</code> , <code>sub_c</code> , and <code>sub_c</code> are the explicit subcircuit names to be excluded.

---

Argument	Description
<pre>-numvd <i>num_val</i></pre>	<p>Limits the number of violations per device that are reported. The default is 1.</p> <p>To print all violations, specify the <code>all</code> keyword. To turn off violations, specify 0.</p>
<pre>-check <i>real_expression</i> -min <i>min_value</i> -max <i>max_value</i> [-tth <i>tval</i>]   <i>logic_expression</i> [-logic <i>safe</i>   <i>violation</i>] [-tth <i>tval</i>]</pre>	<p>You must provide a constraint expression, that can be a logic expression or a real-value expression (a real value expression can also be an arithmetic expression). If the expression contains a logical operator, it is considered to be a logic expression.</p> <p>Any logic expression that evaluates to logical false (0) for a period longer than the <i>tval</i> is reported as a violation. You can change it with the <code>-logic</code> option. The default for <code>-logic</code> is set to <code>safe</code>. For information about the <code>safe</code> and <code>violation</code> keywords, see <a href="#">Table 21</a>.</p> <p>A constraint that is a real-value expression is reported as a violation if it is outside of the safe operating range specified by <code>-min</code> and <code>-max</code> arguments.</p> <p>To conform to Tcl conventions, enclose the constraint expression in double quotes or braces if it contains any white space.</p> <p>This expression is defined as a single group for the Tcl parser. This means that it should not contain white space or be grouped with the Tcl grouping operators, double quotes (""), or braces ({}).</p>
<pre>-twindow {<i>tstart</i> <i>tstop</i>} <i>tstart</i> <i>tstop</i> -tstep <i>tstep_val</i></pre>	<p>If you specify this argument, CustomSim performs the <code>cck_soa</code> check within the time window defined by <i>tstart</i> <i>tstop</i>{<i>tstart</i> <i>tstop</i>}. The <i>tstart</i> and <i>tstop</i> must come in pairs, except for the final window where: if <i>tstop</i> is not specified, it is assumed to be the end of the simulation. The final <i>tstop</i> can be the <code>end</code> or <code>END</code> keyword.</p>
<pre>-tstep <i>tstep_val</i></pre>	<p>If you specify this argument, the <code>cck_soa</code> check is performed at discrete time points, which are integer multiples of <i>tstep_val</i>. You can use this argument with the <code>-twindow</code> argument.</p>

## Chapter 4: CustomSim Dynamic CircuitCheck

cck\_soa

Argument	Description
<code>-flush auto auto2</code>	Specify <code>auto</code> to report violations according to all of the following criteria: <ul style="list-style-type: none"><li>▪ Report violations as they occur.</li><li>▪ Report violations if the violation duration satisfies the time period specified by the <code>cck_soa</code> command constraint expression and the violation state change.</li></ul> Specify <code>auto2</code> to report violations according to all of the following criteria: <ul style="list-style-type: none"><li>▪ Report violations as they occur.</li><li>▪ Report violations if the violation duration satisfies the time period specified by the <code>cck_soa</code> command constraint expression.</li></ul>
<code>-report report_type {report_type}</code>	Enables the following additional information to be reported for each device: <ul style="list-style-type: none"><li>▪ <code>wl</code> specifies to report length and width information.</li><li>▪ <code>conn</code> specifies to report connectivity information.</li><li>▪ <code>subinfo</code> specifies to report the subcircuit hierarchy information.</li><li>▪ <code>operand</code> specifies to report the atomic values of the logical expression. For example, when you specify <code>(Vdb&gt;3&amp;&amp;Vgs&gt;0.5)</code>, <code>operand</code> reports the peak values of Vdb and Vgs at the period of the violation.</li></ul>
<code>-numd num_value</code>	Limits the total number of violated devices reported. The default is 300. To print all violated devices, specify the <code>all</code> . To turn off violation reporting, specify 0.
<code>-filteralert 0/1</code>	Specifies to issue the <code>zero scoping error message</code> if there are no devices that match with the user specified scoping rule. This can be caused by an unintended typo or the matched device might be excluded by the <code>-except_inst</code> or the <code>-except_subckt</code> expression in the same <code>cck_soa</code> statement.  By default, ( <code>-filteralert 1</code> ) the command considers such scenario as an error and therefore displays the <code>zero scoping error message</code> and terminates the simulation. If you specify <code>-filteralert 0</code> , only the <code>zero scoping error message</code> is displayed and the simulation is not terminated.

Argument	Description
<code>-portalert enable_value</code>	Enables a warning or error if the Verilog-A port that is specified in the constraint expression cannot be matched. By default, it is set to 1 to issue an error message and terminate the simulation. To issue a warning message for the statement and continue the simulation, set it to 0.
<code>-limit value</code>	Specifies the hierarchy level down to which the scope is applied. When you specify <code>-subckt</code> , the <code>-limit</code> is relative to where the subcircuit is located in the hierarchy. A value of 0 specifies the top level of the subcircuit. The default is for all levels. This option is only effective when scoping includes wildcard characters.

**Description**

This command checks primitive devices for safe operating area conditions. Violations are reported to an output file, `*.ccksoa`.

`cck_soa` checks the following elements:

- MOSFETs
- BJTs
- JFETs
- Diodes
- Resistors and capacitors
- Verilog-A modules

The constraint expression is a required argument that specifies the check to be performed. You can construct the expression with a single or group of constraint functions (listed in [Table 24](#).) with either arithmetic operators (+, -, \*, /) or logical operators (>, <, >=, <=, &&, ||, ==, !=). It can be a real-value constraint or a logical constraint.

A real-value constraint expression only involves arithmetic operators. The expression return value is a normal decimal real value. A violation is reported if the normal decimal real value is outside a safe operating area specified by `-min` and `-max` arguments. However, a logical constraint expression involves logical operators. The expression evaluation return is either logical true (1) or false (0). A logical expression that evaluates to logical false (0) is reported as a violation.

**Table 24** lists the keywords that are used for accessing common instance conditions.

*Table 24 Constraint Functions*

Constraint Function	Description
I	Current of a MOSFET, JFET, BJT, DIODE, RESISTOR or CAPACITOR
lb	Bulk current of a MOSFET or base current of a BJT
lc	Collector current of a BJT
ld	Drain current of MOSFET or BJT
le	Emitter current of a BJT
lg	Gate current of a MOSFET or JFET
ls	Source current of a MOSFET, BJT, or JFET
Vb	Bulk voltage of a MOSFET or base voltage of a BJT
Vbc	Base/collector voltage difference of a BJT ( $V_b - V_c$ )
Vcb	Collector/base voltage difference of a BJT ( $V_c - V_b$ )
Vbd	Bulk/drain voltage difference of a MOSFET or JFET ( $V_b - V_d$ )
Vdb	Drain/bulk voltage difference of a MOSFET or JFET ( $V_d - V_b$ )
Vbe	Base/emitter voltage difference of a BJT ( $V_b - V_e$ )
Veb	Emitter/base voltage difference of a BJT ( $V_e - V_b$ )
Vbs	Bulk/source voltage difference of a MOSFET or base/source voltage difference of a BJT ( $V_b - V_s$ )
Vsb	Source/bulk voltage difference of a MOSFET or source/base voltage difference of a BJT ( $V_s - V_b$ )
Vc	Collector voltage of a BJT
Vce	Collector/emitter voltage difference of a BJT ( $V_c - V_e$ )

Table 24 Constraint Functions

Constraint Function	Description
Vec	Emitter/collector voltage difference of a BJT ( $V_e - V_c$ )
Vcs	Collector/source voltage difference of a BJT ( $V_c - V_s$ )
Vsc	Source/collector voltage difference of a BJT ( $V_s - V_c$ )
Vd	Drain voltage of a MOSFET or JFET
Vdip	Bias voltage of DIODE, RESISTOR, or CAPACITOR
Vds	Drain/source voltage difference of a MOSFET or JFET ( $V_d - V_s$ )
Vsd	Source/drain voltage difference of a MOSFET or JFET ( $V_s - V_d$ )
Ve	Emitter voltage of a BJT
Ves	Emitter/source voltage difference of a BJT ( $V_e - V_s$ )
Vse	Source/emitter voltage difference of a BJT ( $V_s - V_e$ )
Vse	Source/emitter voltage difference of a BJT ( $V_s - V_e$ )
Vg	Gate voltage of a MOSFET or JFET
Vgb	Gate/bulk voltage difference of a MOSFET or JFET ( $V_g - V_b$ )
Vbg	Bulk/gate voltage difference of a MOSFET or JFET ( $V_b - V_g$ )
Vgd	Gate/drain voltage difference of a MOSFET or JFET ( $V_g - V_d$ )
Vdg	Drain/gate voltage difference of a MOSFET or JFET ( $V_d - V_g$ )
Vgs	Gate/source voltage difference of a MOSFET or JFET ( $V_g - V_s$ )
Vsg	Gate/source voltage difference of a MOSFET or JFET ( $V_s - V_g$ )
Vneg	Cathode voltage of a DIODE or low-voltage terminal voltage of RESISTOR or CAPACITOR

**Chapter 4: CustomSim Dynamic CircuitCheck**  
cck\_soa
*Table 24 Constraint Functions*

<b>Constraint Function</b>	<b>Description</b>
Vpos	Anode voltage of a DIODE or high-voltage terminal voltage of RESISTOR or CAPACITOR
Vs	Source voltage of a MOSFET, JFET, or BJT
W	Width of a MOSFET
L	Length of a MOSFET
V(port)	Port voltage of Verilog-A Module

**Table 25** lists the mathematical functions that are supported to construct the constraint expressions.

*Table 25 Mathematical Functions*

<b>Built-In Function</b>	<b>Description</b>
abs(x)	Returns the absolute value of x:  x
ceil(x)	Returns the integer that is greater than or equal to the value of x
exp(x)	Returns e, raised to the power of the value of x: e <sup>x</sup>
floor(x)	Returns the integer that is less than or equal to the value of x
log(x)	Returns the natural logarithm of the absolute value of x with the sign of x: (sign of x)log( x )
log10(x)	Returns the base 10 logarithm of the absolute value of x with the sign of x: (sign of x)log10( x )
pow(x,y)	Returns the value of x raised to the power of the value of y
sqrt(x)	Returns the square root of the absolute value of x: sqrt(-x) = -sqrt( x )

To conform to the Tcl convention, enclose the constraint expression in double quotes or braces if it contains any white space.

You can use `cck_soa` to scope to specific instances using the `-model`, `-inst`, `-subckt`, `-except_inst`, `-expect_subckt`, and `-instparam` arguments. You can use wildcards only with the `-inst` and `-except_inst` arguments. Wildcard characters are not supported for the `-model`, `-subckt`, and `-except_subckt` arguments. If you use the `-subckt` argument with the `-inst` argument or the `-model` argument, the named instances or named models are local to the specified subcircuits.

The rule between `-except_subckt` and `-except_inst` arguments accumulates in a logical OR. If you use the `-except_subckt` argument with `-except_inst`, the checks exclude all instances inside the named subcircuit by the `-except_subckt` argument and all the named instances by `-except_inst`.

You can specify resistors, capacitors, and diodes for the `-model` argument, by using the reserved keywords `all_resistors`, `all_capacitors`, and `all_diodes` respectively. All resistors are checked if the `-model all_resistors` argument is specified; all capacitors are checked if the `-model all_capacitors` argument is specified; and all diodes are checked if `-model all_diodes` argument is specified. The names of the resistors, capacitors, and diodes models can also be specified in the `in` the `-model` argument if only certain models are to be checked.

The Verilog-A module also is supported by specifying the names of the Verilog-A module in the `-model` argument. It uses `v` (portname) to monitor the port voltage of the Verilog-A module in the `-check` argument.

If you specify the `-instparam` argument, the check is limited to only MOS devices whose width and length meet the instance parameter expression. You can construct the instance parameter expression with a single or a group of instance parameter functions with logical operators (`>`, `<`, `>=`, `<=`, `&&`, `||`, `==`, `!=`). These two instance parameter functions can be used to create an instance parameter expression.

*Table 26 Instance Parameters*

Instance Parameter Function	Description
L	Length of a MOS device
W	Width of a MOS device

When you specify the `-flush` argument, on-the-fly auto-reporting is enabled. Each `cck_soa` command for which auto-flushing is enabled writes its output to

**Chapter 4: CustomSim Dynamic CircuitCheck**  
**cck\_soa**

a unique file. These files are named as *prefix.ccksoa.label*, where *prefix* is the input netlist filename *prefix*, or the name specified with *-o* on the command line. The label is the command label specified by *-label* argument of that command. Each file is updated as violations are detected.

When you specify the *-report* argument, it triggers additional information to be reported in the output file. The *cck\_soa* command supports reporting the width and length and connectivity information for each device that violates the check. You can enable the

*-report wl* argument and connectivity information by using  
*-report conn*. The following example shows a sample report:

Dev #1: N1

Attributes: model=NMOS\_test w=10u l=0.5u

Connectivity:

D: net052

G: net17

S: 0

G: 0

**Examples***Example 90 Constraint Examples*

```
cck_soa vdschk -check vds -min 0.3 -max 0.5
```

Checks for the voltage difference between drain and source of MOSFET and JFET. A violation is reported if the voltage difference is out of safe operating range of 0.3 V – 0.5 V.

*Example 91 Logical Expression*

```
cck_soa logcheck -check "vds>0.3 && vds<0.5 && vgs>0 && vgs<0.5"
```

Shows a logical expression. If *vds* > 0.3 and *vds* < 0.5, and *vgs* > 0 and *vgs* < 0.5, the return value is logical true (1), and no violation is reported. Otherwise a violation is reported.

*Example 92 Logical Violation*

```
cck_soa vgscheck -check "vgs<0" -logic violation
```

Shows a logical violation. If *vgs* < 0, the return value is logical true (1), and a violation is reported.

*Example 93 Scoping Examples -1*

```
cck_soa subcktonly -subckt nchmos_mac -check vgs -min 0 -max 3 ...
```

Applies the check to all instances and devices of nchmos\_mac subcircuit.

*Example 94 Scoping Examples -2*

```
cck_soa subinst -subckt mysub -inst x1 -check vgs -min 0 -max 3 ...  
.subckt mysub port1 port2 port3
```

```
X1 port1 port2 somesub  
X2 port3 port4 somesub  
...  
.ends
```

Applies the check only to the x1 instance of the mysub subcircuit. All other instances are excluded from the check.

*Example 95 MOS Devices*

```
cck_soa instparam -model nch -instparam { l < 0.1u } \  
-check vgs -min -1.0 -max 1.0
```

Applies the check to all MOS devices named, nch and having length less than 0.1 um.

*Example 96 Exclude Instances*

```
cck_soa except -model nch \  
-except_inst x1.x2.* -except_subckt dummy_cell -check "vgs < 0.6"
```

In this example, you can exclude instances from scoping using the –except\_inst and –except\_subckt arguments. When the –except\_inst argument and –except\_subckt argument are used in the same command, the checks exclude the instances specified by the –except\_inst argument and all the instances inside the subcircuit specified by –except\_subckt. This example applies the check to all the devices with model name nch, except for those instances of x1.x2 instance and all the instances inside the dummy\_cell subcircuit.

*Example 97 Resistor Devices*

```
cck_soa res_model -model all_resistors -check { vpos-vneg > 2 }
```

Applies the check to all resistor devices.

**Chapter 4: CustomSim Dynamic CircuitCheck**

cck\_substrate

**Example 98 Verilog- A**

```
cck_soa instparam -model mos_va -check { v(port1) > 10 }
```

Applies the check to Verilog-A module named `mos_va`.

**Example 99 Real-Value, Logic, Time Window, and Discrete Points Examples**

```
cck_soa simple_check -model nch -check vgs -min 0 -max 0.6
```

Shows a real-value check.

**Example 100 Logic Check**

```
cck_soa logic_check -model nch -check {VGS > 0 && VGS < 0.6}
```

```
cck_soa logic_check2 -model nch \
-check "(VGS>0) && (VGS<0.6) && (ID<200n)"
```

Shows a logic check.

**Example 101 Real-value check with Time Window**

```
cck_soa simple_check_3n -model nch -check vgs \
-min 0 -max 0.6 -tth 3n\
```

```
cck_soa windows -model nch -check vgs -min 0 -max 0.6 \
-twindow 1u 2u 3u 4u
```

Shows a real-value check with time window.

**Example 102 Discrete Point Check**

```
cck_soa step -model nch -check vgs -min 0 -max 0.6 -tstep 500n
```

```
cck_soa window_step -model nch -check vgs -min 0 -max 0.6 \
-twindow 500n 5u -tstep 200n
```

Shows the discrete points check.

---

**cck\_substrate**

Checks for a forward-bias bulk condition on MOS devices.

**Syntax**

```
cck_substrate -label lname [scope] [-numv num] [twindow] [-vth value] [-ith value] [-tth value] [-flush auto] [report]
```

```

scope ::= [model] [inst] [subckt] [except_i] [except_s]

model ::= -model model_name {model_name}

inst ::= -inst inst_pattern {inst_pattern}

subckt ::= -subckt subckt_name {subckt_name}

except_i ::= -except_inst inst_pattern {inst_pattern}

except_s ::= -except_subckt subckt_name[.inst_pattern]
           {subckt_name[.inst_pattern]}

twindow ::= [-twindow {tstart tstop} tstart [tstop]]
           [-tstep tstep_val]

report ::= -report report_type {report_type}

report_type ::= wl | conn

```

---

Argument	Description
-label label_name	Specifies the label name that appears in the report file, which makes it easier to search the report. If on-the-fly reporting is enabled, the label forms part of the report filename.
-model model_name {model_name}	Scopes to devices of the named models.
-inst inst_pattern {inst_pattern}	Scopes to the named instances. You can use wildcards in instance names as specified by the CustomSim set_wildcard_rule command. Wildcard behavior is determined by the set_wildcard_rule command.
-subckt subckt_name {subckt_name}	Scopes to instances of the named subcircuits. When you use the -subckt argument, the -model and -inst arguments become local to the named subcircuits.

## Chapter 4: CustomSim Dynamic CircuitCheck

cck\_substrate

Argument	Description
<code>-except_inst inst_name {sub_name}</code>	Excludes all instances of the named subcircuits from the <code>cck_substrate</code> command scope. Any subcircuit whose name matches the pattern is excluded from the command scope. Wildcards are not supported.
<code>-except_subckt sub_name[.inst_ pattern] {sub_name}[.inst_ pattern]</code>	Excludes all instances of the named subcircuits from the <code>cck_substrate</code> command scope. Any subcircuit whose name matches the pattern is excluded from the scope. You can also exclude only the named instances of the subcircuit. Wildcards are not supported in subcircuit names, but are supported in the <code>inst_pattern</code> .
<code>-numv num_val</code>	Limits the total number of violations reported. To print all violations, specify the <code>all</code> keyword. To turn off violation report checking, specify <code>0</code> . The default is <code>300</code> .
<code>-twindow {tstart tstop} tstart tstop -tstep tstep_val</code>	If you specify this argument, CustomSim performs the <code>cck_substrate</code> check within the time window defined by <code>tstart tstop</code> { <code>tstart tstop</code> }. The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where: if <code>tstop</code> is not specified, it is assumed to be the end of the simulation. The final <code>tstop</code> can also be the <code>end</code> or <code>END</code> keyword.
<code>-vth vth</code>	Specifies that the bulk must be forward biased by the specified amount before a violation is reported. The default is <code>0.5 V</code> .
<code>-ith ith</code>	Specifies that the bulk current must exceed the specified value for a violation to be reported. Note, if you want to apply the bulk current as a substrate violation criteria, it is recommended to set higher accuracy model level (such as model level 6 or 7) in order to obtain an accurate bulk current value. There is no default value for this control option. If this option is not specified, the command will disable the bulk current check.
<code>-tth value</code>	Specifies the time duration threshold value to be reported. The violation must last longer than the specified time value.

Argument	Description
<code>-tstep <i>tstep_val</i></code>	Specifies that the <code>cck_substrate</code> check is performed at discrete time points that are integer multiples of <code>tstep_val</code> . This argument can be used with the <code>-twindow</code> argument.
<code>-flush auto</code>	Reports the violations that are reported after the completion of the simulation, if you do not specify this argument. To enable Auto-flushing, specify the <code>auto</code> keyword that specifies to report violations as they occur on-the-fly in the simulation.
<code>-report <i>report_type</i> {<i>report_type</i>}</code>	Enables additional information to be reported for each device. To enable the Width and length information, specify the <code>wl</code> keyword. To enable the Connectivity information, specify the <code>conn</code> keyword.

### Description

The `cck_substrate` command checks for a forward-bias bulk condition on MOS devices. A violation is reported when a MOSFET meets the following conditions:

$$\begin{cases} pmos, v_s - v_b > v_t \text{ or } v_d - v_b > v_t \\ nmos, v_b - v_s > v_t \text{ or } v_b - v_d > v_t \end{cases}$$

Violations are reported to an output file, `*.substrate`.

You can use the `cck_substrate` to scope to specific instances using the `-model`, `-inst`, `-subckt`, `-except_inst`, `-expect_subckt`, and `-instparam` arguments. You can use wildcards only with the `-inst` and `-except_inst` arguments. Wildcard characters are not supported for the `-model`, `-subckt`, and `-except_subckt` arguments. If you use the `-subckt` argument with the `-inst` argument or the `-model` argument, the named instances or named models are local to the specified subcircuits.

The rule between `-except_subckt` and `-except_inst` arguments accumulates in a logical OR. If you use the `-except_subckt` argument with `-except_inst`, the checks exclude all the instances inside the named subcircuit by the `-except_subckt` argument and all the named instances by the `-except_inst` argument.

When you specify the `-flush` argument, on-the-fly auto-reporting is enabled. Each `cck_substrate` command for which auto-flushing is enabled writes its

**Chapter 4: CustomSim Dynamic CircuitCheck**

cck\_substrate

output to a unique file. These files are named as

*prefix.ccksubstrate.label*, where *prefix* is the input netlist filename *prefix*, or the name specified with *-o* on the command line. The label is the command label specified by *-label* argument of that command. Each file is updated as violations are detected.

When you specify the *-report* argument, it triggers additional information to be reported in the output file. For each *cck\_substrate* command in which *-report wl* is specified, width and length information is reported for each device that violates the check as shown by:

Dev #1: mp

Attributes: model=p w=1u l=0.5u

When you specify the *-report conn* argument, connectivity information is reported for each device as follows:

Dev #1: xm1.m1

Attributes: type=PMOS

Connectivity:

D: 0

G: VDD

S: VDD

B: BODY

## Examples

### *Example 103 Scoping Examples*

```
cck_substrate subcktonly -subckt nchmos_mac ...
```

Applies the check to all instances and devices of *nchmos\_mac* subcircuit.

```
cck_substrate except -model nch \
-except_inst x1.x2.* -except_subckt dummy_cell ...
```

When you use *-except\_inst* and *-except\_subckt* in the same command, the checks exclude the instances specified by *-except\_inst* and all the instances inside the subcircuit specified by *-except\_subckt*. The previous example applies the check to all the devices with model name *nch*, except for those instances of *x1.x2* and all the instances inside *dummy\_cell* subcircuit.

*Example 104 MOS Checking Examples*

```
cck_substrate -label nmos_bulk_chk -model nch
```

Checks only MOS instances of the model nch for forward biased bulk conditions. The check is performed from t = 0 to the end of the simulation.

```
cck_substrate -label bulk_check_550_600 -twindow 550n 600n
```

Checks all MOS instances for forward biased bulk conditions only during the time window 550– 600 ns.

---

## cck\_toggle\_count

Checks toggling activity on the specified nodes.

### Syntax

```
cck_toggle_count -label lname [report_type] [scope]
    [twindow] [-flush value%] [-report_most_toggle value]
    [-report_zero_toggle value| all]
    [-report_multi_toggle period [num_toggle]]
```

*report\_type* ::= *basic\_report* | *detail\_report*

*basic\_report* ::= -enable\_detail 0|no|false| off [-vth value]

*detail\_report* ::= -enable\_detail 1|yes|true|on
 [-loth|-vol value] [-hith|-voh value]

*scope* ::= [*node*] [*subckt*] [*except\_n*] [*except\_s*]

*node* ::= -node *node\_pattern* {*node\_pattern*}

*subckt* ::= -subckt *subckt\_name* {*subckt\_name*}

*except\_n* ::= -except\_node *node\_pattern* {*node\_pattern*}

*except\_s* ::= -except\_subckt *subckt\_name*[.*node\_pattern*]
 {*subckt\_name*[.*node\_pattern*]}

*twindow* ::= [-twindow {*tstart* *tstop*} *tstart* [*tstop*]]

## Chapter 4: CustomSim Dynamic CircuitCheck

cck\_toggle\_count

Argument	Description
<code>-label <i>label_name</i></code>	Specifies the label name that appears in the report file, which makes it easier to search the report. If on-the-fly reporting is enabled, the label forms part of the report filename.
<code>-enable_detail enable_value</code>	Enables the type of report to be generated. A basic report is generated by default, or when the argument is set to 0 (off). A detail report is generated when the argument is set to 1 (on).
<code>-vth <i>vth</i></code>	Specifies the voltage reference if the basic report is enabled. The default is $(V_{max}+V_{min})*0.5$ . $V_{max}$ and $V_{min}$ are different among various voltage domains.
<code>-loth   -vol <i>vlth</i></code>	Specifies the low voltage threshold if the detail report is enabled. The default follows the value specified in the <code>set_logic_threshold</code> command.
<code>-hith   -voh <i>vhth</i></code>	Specifies the high voltage threshold if the detail report is enabled. The default follows the value specified in <code>set_logic_threshold</code> command.
<code>-node <i>node_pattern</i> {<i>node_pattern</i>}</code>	Scopes to the named nodes. You can use wildcard characters in the node pattern. Wildcard behavior is determined by the <code>set_wildcard_rule</code> command.
<code>-subckt <i>subckt_name</i> {<i>subckt_name</i>}</code>	Scopes to nodes of the named subcircuits. When you specify <code>-subckt</code> , node arguments become local to the named subcircuits. Wildcard characters are not supported in subcircuit names.
<code>-except_node <i>node_pattern</i> {<i>node_pattern</i>}</code>	Excludes nodes from the command scope. Any node whose name matches the pattern is excluded from the command scope. You can use wildcard characters in the node patterns.

Argument	Description
<code>-except_subckt sub_name[.node_ pattern] {sub_name} [.node_ pattern]</code>	Excludes all instances of the named subcircuits from the <code>cck_toggle_count</code> command scope. Any subcircuit whose name matches the pattern is excluded from the scope. You can also exclude only the named instances of the subcircuit. Wildcards are not supported in subcircuit names but are supported in the <code>node_pattern</code> .
<code>-twindow {tstart tstop} tstart [tstop]</code>	Limits checking to the specified time windows. The default for <code>tstart</code> is 0 and <code>tstop</code> is the end of transient simulation time.
<code>-flush value%</code>	Reports violations after the completion of the simulation if you do not specify this argument. Flushing is enabled by providing the frequency in a percentage value to report violations.
<code>-report_most_toggle value</code>	Specifies the number of the most toggling nodes to be reported.
<code>-report_zero_toggle value   all</code>	Specifies the number of zero toggling node to be reported. To report all zero toggling nodes, specify the <code>all</code> keyword.
<code>-report_multi_toggle period [num_toggle]</code>	Reports nodes that toggle more than <code>num_toggle</code> times within a specified period.

### Description

You can use `cck_toggle_count` to scope to specific instances using the `-node`, `-subckt`, `-except_node`, and `-expect_subckt` arguments. You can use wildcards only with the `-node` and `-except_node` arguments. Wildcard characters are not supported for the `-subckt` and `-except_subckt` arguments. If you use the `-subckt` argument with the `-node` argument, the named nodes are local to the specified subcircuits.

The rule between `-except_subckt` and `-except_node` arguments accumulates in a logical OR. If you use the `-except_subckt` argument with `-except_node`, the checks exclude all the nodes inside the named subcircuit by the `-except_subckt` argument and all the named nodes by the `-except_node` argument.

**Chapter 4: CustomSim Dynamic CircuitCheck****cck\_toggle\_count**

You can use the `cck_toggle_count` command to create a basic report or a detail report. A basic report uses the single voltage threshold to determine toggling. It contains only the number of toggles per node. A detail report uses two voltage thresholds to determine toggling and contains the following information:

- Rising toggle
- Falling toggle
- Average overshoot
- Average undershoot
- Rise time (max/min/avg)
- Fall time (max/min/avg)
- Probability high/low

The following three arguments (`-vth`, `-loth`, and `-hith`) are used to collect statistic data about toggling activities as follows:

- Most active toggling nodes
- Zero toggling nodes
- Nodes of specified number of toggles within a specified period

When you specify the `-flush` argument, on-the-fly auto reporting is enabled. Each `cck_toggle_count` command for which auto flushing is enabled writes its output to a unique file. These files are named as `prefix.ccktoggle.label`, where `prefix` is the input netlist filename `prefix`, or the name specified with `-o` on the command line. The label is the command label specified by `-label` argument of that command. Each file is updated as violations are detected.

As shown in [Figure 8 on page 345](#), [Figure 9 on page 345](#), and [Figure 10 on page 346](#), the calculation of the toggle count depends on the number of thresholds. [Figure 8](#) illustrates that the basic report uses a single threshold:

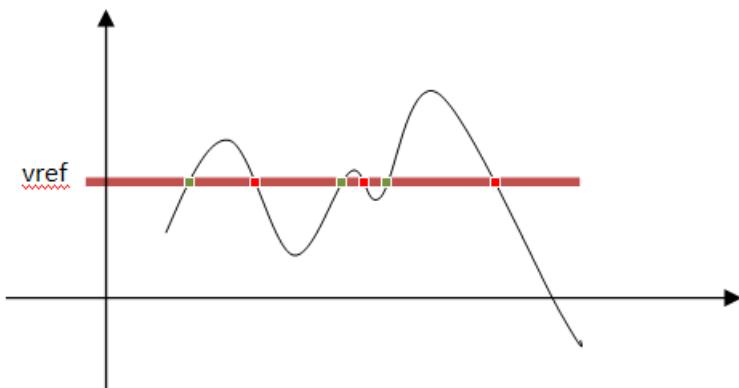


Figure 8 Single Threshold

Figure 9 illustrates that the detail report uses two thresholds:

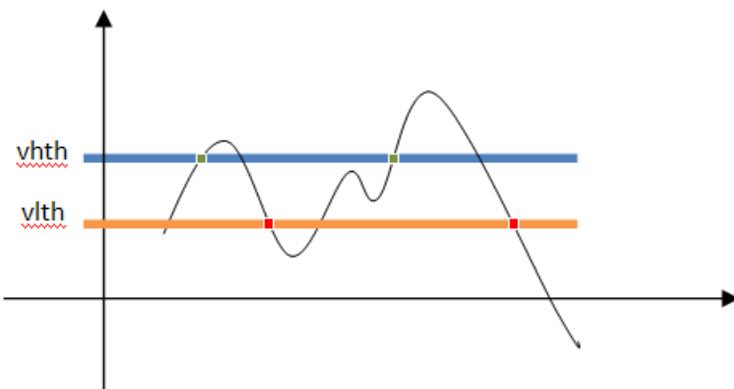


Figure 9 Two Thresholds

In Figure 10, the colored areas illustrate undershoots (red) and overshoots (green). The time average values are obtained from dividing the colored areas by time  $t$ .

## Feedback

### Chapter 4: CustomSim Dynamic CircuitCheck

cck\_toggle\_count

Formula:

$$\text{Overshoot} = \frac{\sum_i \int_{\Delta T_i^o} v_o(t) dt}{T}, \text{ where } v_o(t) = v(t) - v_{hth}, t \in \Delta T_i^o \text{ s.t. } v(t) > v_{hth}$$

$$\text{Undershoot} = \frac{\sum_i \int_{\Delta T_i^u} v_u(t) dt}{T}, \text{ where } v_u(t) = v_{lth} - v(t), t \in \Delta T_i^u \text{ s.t. } v(t) < v_{lth}$$

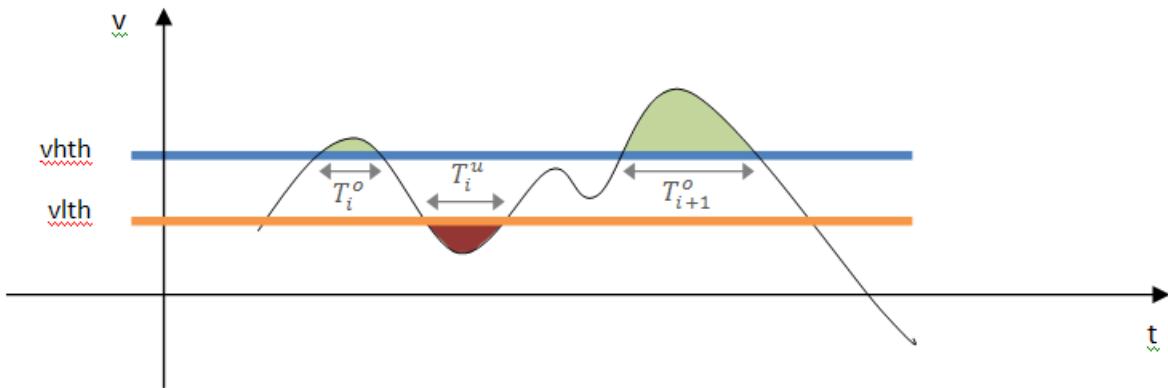


Figure 10 Overshoot and Undershoot

The rise time is the time span of the green area; while the falling time is of the red area. The green areas can be defined by two time points when the voltage is rising, where:

- $t_1$  is the first moment when  $v(t) > v_{lth}$
- $t_2$  is the first moment when  $v(t) > v_{hth}$  after  $t_1$

The red areas are also defined by two time points when the voltage is falling, where:

- $t_1$  is the first moment when  $v(t) < v_{hth}$
- $t_2$  is the first moment when  $v(t) < v_{lth}$  after  $t_1$

See [Figure 11](#).

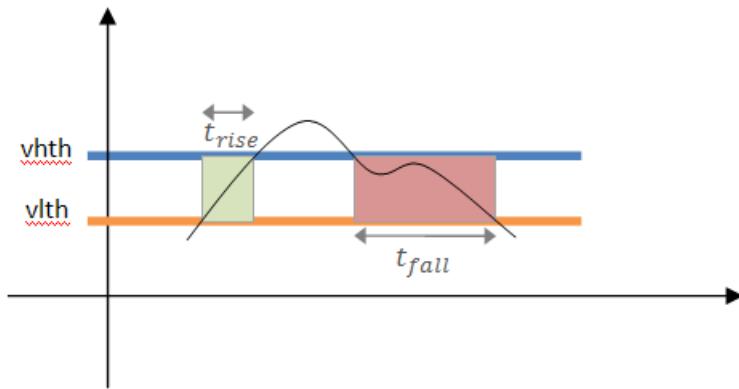


Figure 11 Rising and Falling Times

The probability of high is obtained by dividing the whole simulation time by the total time of the green areas. If the denominator is replaced by the total time of the red areas, then the probability of low is obtained. See [Figure 12](#).

Formula:

$$\text{Prob. of High} = \frac{\sum_i \Delta T_i^o}{T}, \text{ where } t \in \Delta T_i^o \text{ s.t. } v(t) > v_{hth}$$

$$\text{Prob. of Low} = \frac{\sum_i \Delta T_i^u}{T}, \text{ where } t \in \Delta T_i^u \text{ s.t. } v(t) < v_{lth}$$

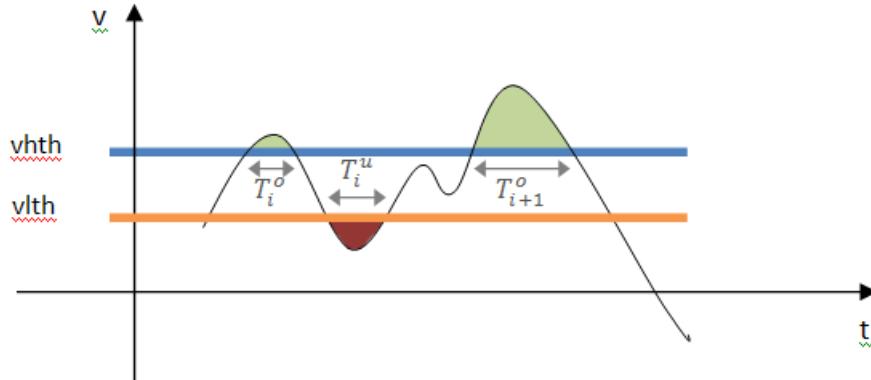
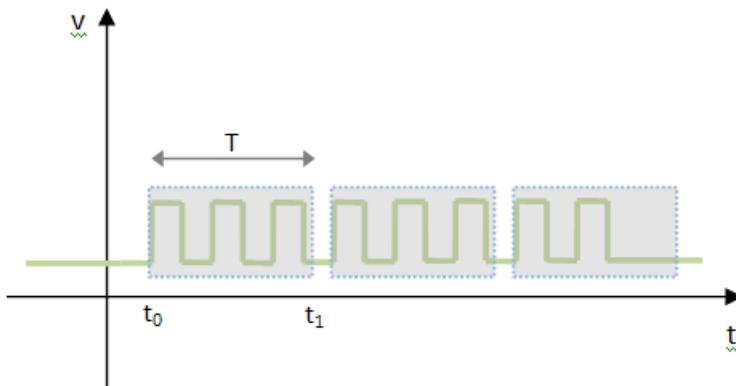


Figure 12 Probability of High and Low

**Chapter 4: CustomSim Dynamic CircuitCheck**

cck\_toggle\_count

Multiple toggling means a node toggles multiple times within a given time period. A time period starts at the point when the first toggle occurs and lasts for a user-specified time. The next time period starts when the next toggle occurs. See [Figure 13](#).



*Figure 13 Multiple Toggling*

### Examples

#### *Example 105*

```
cck_toggle_count -label test1 -node x1.sout -flush 10%
```

Checks the toggling activity on the `x1.sout` node. On-the-fly reporting is enabled and updated with the basic node activity every 10% of the simulation.

#### *Example 106*

```
cck_toggle_count -label test2 -except_node n*
```

Checks the toggling activity on all nodes, except for nodes with names that start with `n`. A basic report is printed at the end of the simulation.

#### *Example 107*

```
cck_toggle_count test4 -detail 1 -subckt dummy -node ain*
```

Checks the toggling activity on all nodes with names that start with `ain` inside the `dummy` subcircuit. A detail report is generated at the end of the simulation.

*Example 108*

```
cck_toggle_count test5 -report_most_toggle 2 -report_zero_toggle
3
```

Checks the toggling activity on all nodes. A basic report is printed to an output file at the end of the simulation with additional information on the two most toggling nodes and three zero toggling nodes.

## **cck\_analog\_pdown**

Detects unwanted currents due to floating MOS gates.

### Syntax

```
cck_analog_pdown [-label] lname when_to_check [path]
[mos_control] [bjt_control] [diode_control] [-numv
numv|all] [-rule 0|1]

when_to_check ::= twindow_periodical_times|specific_times

twindow_periodical_times ::= [-twindow {tstart tstop} tstart
[tstop]] -tstep tstep_val

specific_times ::= -at at_time {at_time}

path ::= [-from_node nname {nname}] [-to_node nname {nname}]

mos_control ::= [-idsth ids_val] [-dvt dvt_val]
[-model_dvt mname mdvt {mname mdvt}]

bjt_control ::= [-vbeth vbe_val]

diode_control ::= [-diode_vth vth_val|-diode_ith ith_val]
```

Argument	Description
-label	Specifies the label name that appears in the report file, which makes it easier to search the report.

## Chapter 4: CustomSim Dynamic CircuitCheck

cck\_analog\_pdown

Argument	Description
<code>-twindow {tstart tstop} tstart [tstop]</code>	Instructs CustomSim to perform the check within the time window defined by <code>tstart tstop</code> . The <code>tstart</code> and <code>tstop</code> must come in pairs, except for the final window where: if <code>tstop</code> is not specified, it is assumed to be the end of the simulation. The final <code>tstop</code> can also be the <code>end</code> or <code>END</code> keyword. You must use this option along with <code>-tstep</code> to perform discrete checking. Continuous checking is not supported in <code>cck_analog_pdown</code> .
<code>-tstep tstep_val</code>	Performs a discrete check at the time points that are integer multiples of <code>tstep_val</code> . You must use this option with the <code>-twindow</code> option. If there is only a <code>-tstep</code> option without a <code>-twindow</code> setting, CustomSim generates an error message and stops the simulation.
<code>-at at_time {at_time}</code>	Performs a discrete check at the specified time. The usage of <code>-tstep</code> and <code>-at</code> is mutually exclusive. If you specify both options, CustomSim generates an error message and stops the simulation.
<code>-from_node nname {nname}</code>	Specifies the starting nodes of a path to be checked, which avoids a long runtime due to many paths. Wildcards are supported. When you use a wildcard character (*), it only matches the nodes that have a direct connection to the voltage source.
<code>-to_node nname {nname}</code>	Specifies the ending nodes of a path to be checked, which avoids a long runtime due to many paths. Wildcards are supported. When you use a wildcard character (*), it only matches the nodes that directly connect to the voltage source.
<code>-idsth ids_val</code>	Specifies the <code>ids</code> threshold for a MOS to be considered conducting. The default is <code>1e-8 A</code> , which is the same as the <code>set_zstate_option</code> command.
<code>-dvt dvt_val</code>	Specifies the delta <code>vt</code> threshold for a MOS to be considered conducting. The default is 0.

---

Argument	Description
<code>-model_dvt mname mdvt {mname mdvt}</code>	Specifies the pairs of model name and its delta vt threshold value. <code>cck_analog_pdown</code> uses its related dvt value of defined model to apply to the checking rule for a MOS.
<code>-vbeth vbe_val</code>	Specifies the <code>vbe</code> threshold for a BJT to be considered conducting. The default is 0.64. For more information, see <a href="#">Table 27</a> and <a href="#">Table 28</a> .
<code>-diode_vth vth_val</code>	Specifies the forward bias threshold for diodes. To be compatible with the conducting criteria of the <code>set_zstate_option</code> command, you must set this value to 0.2.
<code>-diode_ith ith_val</code>	Specifies the diode current threshold to be considered conducting. The default is 1e-8 A.
<code>-numv numv</code>	Limits the maximum number of total violations reported. The default is 300. To print all violations, specify the <code>all</code> keyword.
<code>-rule 0 1</code>	Specifies the conducting rule for analog power-down checking. For more information, see <a href="#">Table 27</a> and <a href="#">Table 28</a> .

---

### Description

Unwanted currents in standby mode are typically checked by simulating the circuit in standby mode and observing the supply current. [cck\\_excess\\_ipath](#) can help with this task. However, unwanted currents due to floating MOS gates cannot be reliably predicted because the voltage values of floating nodes cannot be easily simulated (vector-dependence, very large time constant, and so on). [Figure 14](#) illustrates a situation that typically leads to unwanted current on silicon, which may not be observed in transient simulation. Whether or not the unwanted current through M5 and M6 can be observed depends on the voltage of n3, which cannot be easily simulated. In general, designers want to avoid this type of situation.

The following keywords are used in the `cck_analog_pdown` output file:

## Feedback

### Chapter 4: CustomSim Dynamic CircuitCheck cck\_analog\_pdown

- gateIsHiZ for a MOS element whose gate is a Hi-Z node
- baseIsHiZ for a BJT element whose base is a Hi-Z node
- conducting for a MOS element whose gate is not a Hi-Z node

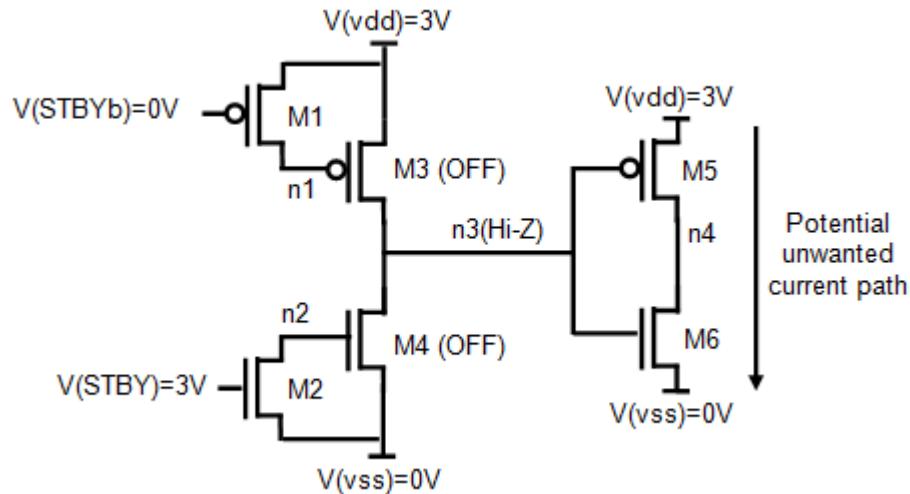


Figure 14 Potential Unwanted Standby Current

The `cck_analog_pdown` command addresses this type of situation by detecting paths constituted not only of conducting elements, but also elements driven by HiZ floating nodes. [Table 27](#) and [Table 28](#) lists the definitions of conducting elements.

*Table 27 Conducting criteria for set\_zstate\_option and check\_node\_zstate*

Device	Rule
NMOS	$V_{gs} > V_{th}$ ( <code>zstate_rule=1</code> )    $I_{ds} > idsth$ ( <code>zstate_rule=2</code> )    $V_g > VDD - 0.1$ ( <code>zstate_rule=3</code> ) The default for <code>idsth</code> is $1e-8A$ . The default rule is 1 and 2.
PMOS	$V_{gs} < V_{th}$ ( <code>zstate_rule=1</code> )    $I_{ds} > idsth$ ( <code>zstate_rule=2</code> )    $V_g < 0.1$ ( <code>zstate_rule=3</code> ) The default for <code>idsth</code> is $1e-8A$ . The default rule is 1 and 2.
NPN	$V_{be} > V_{beth}$ The default for <code>vbeth</code> is 0.64.
PNP	$V_{be} < -V_{beth}$ . The default for <code>Vbeth</code> is 0.64.

**Table 27** Conducting criteria for set\_zstate\_option and check\_node\_zstate

Device	Rule
Diode	$I(diode) > diode\_ith \parallel V(a,c) > diode\_vth$ The default for diode_ith is 1e-8A.
Resistor	Always conducting.
Inductor	Always conducting.
Capacitor	Never conducting.
Verilog-A	Never conducting.
Controlled sources	Always conducting.

**Table 28** Conducting element criteria for cck\_analog\_pdown

Device	rule=1	rule=0
NMOS	$Gate \text{ is a } hz \text{ node} \parallel ((I_{ds} >= idsth \parallel V_g - min(V_d, V_s) >= model->vth - dvt) \&& v(from) >= v(to))$	$(I_{ds} >= idsth \parallel V_g - min(V_d, V_s) >= model->vth - dvt \parallel gate \text{ node is a } hz \text{ node}) \&& v(from) >= v(to)$
PMOS	$Gate \text{ is a } hz \text{ node} \parallel ((I_{ds} >= idsth \parallel max(V_d, V_s) - V_g >= model->vth - dvt) \&& v(from) >= v(to))$	$(I_{ds} >= idsth \parallel Max(V_d, V_s) - V_g >= model->vth - dvt \parallel gate \text{ node is a } hz \text{ node}) \&& v(from) >= v(to)$
BJT(NPN)	$(V_{be} >= V_{beth} \parallel Base \text{ node is a } hz \text{ node}) \&& v(from) >= v(to)$	
BJT(PNP)	$(V_{eb} >= V_{beth} \parallel Base \text{ node is a } hz \text{ node}) \&& v(from) >= v(to)$	
Diode	$(Id >= diode\_ith) \&& (v(from) >= v(to))$	
Resistor	$I > 0 \&& v(from) >= v(to)$	
Other	Never conducting.	

**Chapter 4: CustomSim Dynamic CircuitCheck****cck\_resistive\_path\_report**

**Note:** Due to the embedded CustomSim RC optimization techniques, some resistors in the RC network are trimmed. This impacts the quality of the report generated by the dynamic CCK path analysis commands, such as `cck_analog_pdown` and `cck_excess_ipath` as the expected resistors along the reported path are unavailable.

The command, `cck_resistive_path_report` can be applied together with the dynamic CCK path analysis commands to retain the original resistors and to ensure quality of the report. For more information see, [cck\\_resistive\\_path\\_report on page 354](#)

---

## **cck\_resistive\_path\_report**

Due to embedded CSIM RC optimization methods some resistors in RC network are trimmed. This impacts the quality of the report generated by the dynamic CCK path analysis commands, `cck_analog_pdown` and `cck_excess_ipath` as the expected resistors along the reported path are not available. The `cck_resistive_path_report` command is used to retain the original resistors thereby ensuring that there is no impact on the performance.

### **Syntax**

```
cck_resistive_path_report -level <value>
```

Argument	Description
-level <value>	<p>Specify one of the following value:</p> <ul style="list-style-type: none"><li>▪ 0: To skip current probing on resistive nodes and their connected resistors along the path. The default is 0.</li><li>▪ 1: To enable current probing on resistive nodes and related resistors. The RC optimization algorithms are applied to the resistive networks monitored by the two CCK commands. Due to the RC optimization, some resistors might be missing from the report.</li><li>▪ 2: To enable current probing on resistive nodes and related resistors. In this case the RC optimization algorithms are applied to a lesser extent than the previous setting (-level 1), to the resistive network monitored by the two CCK commands. Due to the RC optimization, some resistors in the power net, might be missing from the report.</li><li>▪ 3: To enable current probing on resistive nodes and related resistors. In this case the RC optimization algorithms are applied to a lesser extent than the previous setting (-level 2), to the resistive network monitored by the two CCK commands. All the resistors in the path are retained.</li></ul>

**Description**

In the following example,

```
cck_resistive_path_report -level 1
```

In this example, RC optimization is applied to the resistive network monitored by the two CCK commands, cck\_analog\_pdown and cck\_excess\_ipath. The two commands will report the resistive network as a reported path, but some resistors will be missed from the report.

## **Feedback**

**Chapter 4: CustomSim Dynamic CircuitCheck**  
cck\_resistive\_path\_report

## Converting the WDF Waveform Output File Format

---

The convert2out utility provide a post-processing capability to convert the specified signals from the .wdf waveform format to the .out and tabular waveform formats.

---

### Using the convert2out Utility

**Note:** The convert2out syntax arguments are order-dependent.

---

#### **convert2out**

Converts .wdf or .fsdb waveform files to .out format.

##### **Syntax**

```
convert2out -i input_filename { [-s signal_filename]  
    [-o output_dir/[output_filename]] } [-compress z|gz|0]
```

or

```
convert2out [-h|-help] [-v|-version]
```

---

Argument	Description
-i <i>input_filename</i>	Specifies the name of the input file in .wdf or .fsdb format. The convert2out utility automatically determines the input file format.

---

**Appendix A: Converting the WDF Waveform Output File Format**

Using the convert2out Utility

Argument	Description
<code>-s signal_filename</code>	<p>Specifies the signal file names. Each signal file contains a list of signals to be converted. By default, all the signals in the input file are converted. <a href="#">Table 29</a> describes the signal file format.</p> <p>This argument is order-dependent. All signals specified in a signal file are converted in an output file specified by the <code>-o</code> argument. If you do not specify a <code>-o</code> argument, the default file name is:</p> <p><code>input_filename.signal_filename.out</code>.</p> <p>You can specify wildcards in the signal names. See the Using Wildcards section for the supported patterns. The "*" wildcard character follows the rule defined by the <a href="#">setWildcardRule</a> command. You can specify one <a href="#">setWildcardRule</a> command in a signal file.</p> <p>You can use the <code>setSimCase</code> command to control case sensitivity:</p> <pre>setSimCase [-case upper lower sensitive] upper specifies to convert all nodes to upper case in the .out file. lower specifies to convert all nodes to lower case in the .out file. sensitive specifies to keep the same case as in the FSDB or WDF waveform file.</pre> <p>The default behavior is to output the same case as in the FSDB or WDF waveform file. Note that you must specify <code>setSimCase</code> and <a href="#">setWildcardRule</a> before the signal list in the signal file.</p>

---

Argument	Description
<code>-o output_dir/ output_filename</code>	<p>Specifies the name of the output directory and output file name. The output file contains the signals specified in the signal file in .out waveform format. Note that you cannot repeat the same file name in one convert2out command. The default output file name is: <i>input_filename.signal_filename.out</i>.</p> <p>The default of output directory is the current running directory. If the specified output directory does not exist, convert2out creates it.</p>
	<p>This argument is order-dependent. The output file only contains the list of signals in the signal file specified by <code>-s</code> in the previous argument. If you do not specify <code>-s</code> in the previous argument, the default rule of <code>-s</code> is applied, which means all signals are converted.</p>
<code>-compress z gz 0</code>	<p>Specifies the type of compression applied to the .out file:</p> <ul style="list-style-type: none"> <li>■ <code>z</code> enables compression with the UNIX compress utility.</li> <li>■ <code>gz</code> enables compression with the GNU gzip utility.</li> <li>■ <code>0</code> disables compression.</li> </ul> <p>Compression applies to all output files and cannot be used for individual files.</p>
<code>-help</code>	Prints help information.
<code>-version</code>	Prints version information.

---

### Description

All of the specified signals to be converted must be presented as they exist in the waveform file. The convert2out utility does not support arithmetic functions. Any necessary arithmetic functions need to be done in the waveform viewer tool.

[Table 29](#) shows the signal file format.

*Table 29 Signal File Format*

---

Signal Access Function	Description
<code>v (node)</code>	Voltage of a node.

---

**Appendix A: Converting the WDF Waveform Output File Format**

Using the convert2out Utility

*Table 29 Signal File Format*

Signal Access Function	Description
<code>vn(instance)</code>	Voltage of a terminal <i>n</i> of an instance where <i>n</i> is an integer greater than or equal to 1; or d, g, s, b for MOSFET and c, b, e, s for BJT.
<code>v(node1, node2)</code>	Expression of the difference voltage of two nodes. The utilities do not do any arithmetic calculations. <code>V(node1, node2)</code> must be presented in the original waveform file already.
<code>i(instance)</code>	Current flowing into the first terminal of an instance.
<code>in(instance)</code>	Current flowing into an instance at terminal <i>n</i> where <i>n</i> is an integer greater or equal to 1; or d, g, s, b for MOSFET and c, b, e, s for BJT.
<code>isub(subckt_port) or x(subckt_port)</code>	Total current flowing into the subcircuit port.
Parameter	Expression of a parameter.

**Note:** The node, instance, subckt\_port and parameter functions are case-sensitive. The `v()`, `vn()`, `i()`, `in()`, `isub()`, and `x()` functions are always case-insensitive.

Here is a sample signal file:

```
V(1)
V(sampling)
V(x1.sout)
I(VDD)
I(x1.m@2899)
Isup(x1.VDD)
V(1,2)
X(x2.VDD)
I3(x1.m@1233)
Id(x1.m@1245)
testout
```

**Examples***Example 109*

```
convert2out -i waveform.wdf -s sig1 -o out1.out -s sig2 -o out2.out
```

**Appendix A: Converting the WDF Waveform Output File Format**  
Using the convert2out Utility

Converts the signals in the `sig1` file to an `out1.out` file and the signals in the `sig2` file to an `out2.out` file.

***Example 110***

```
convert2out -i test.fsdb
```

Converts all the signals in the `test.fsdb` input file to a `test.fsdb.out` file.

***Example 111***

```
convert2out -i test.fsdb -s testsig1 -s testsig2 -s testsig3 -o testout.out -s testsig4
```

Converts: all the signals in the `testsig1` to `test.fsdb.testsig1.out`, all the signals in `testsig2` to `test.fsdb.testsig2.out`, all the signals in `testsig3` to `testout.out`, and all the signals in `testsig4` to `test.fsdb.testsig4.out`.

***Example 112***

```
convert2out -i pll.wdf -s sig1 -s sig2 -o sig2.out -o out.out -o out1.out
```

Converts: all the signals in the `sig1` to `pll.wdf.sig1.out`, all the signals in `sig2` to `sig2.out`, all the signals in `pll.wdf` to `out.out`, and all the signals in `pll.wdf` to `out1.out`.

***Example 113***

```
convert2out -i test.fsdb -o out1.out -o out1.out
```

Generates an error messages because two output files have the same name (`out1.out`).

***Example 114***

```
convert2out -i test.fsdb -o TEST/
```

Converts all the signals in the `test.fsdb` file to a `TEST/test.fsdb.out` file.

***Example 115***

```
convert2out -i test.fsdb -o TEST/test.out
```

Converts all the signals in the `test.fsdb` file to a `TEST/test.out` file.

## Feedback

### Appendix A: Converting the WDF Waveform Output File Format

Using the convert2out Utility

#### *Example 116*

```
convert2out -i waveform.wdf -s sig1 -o out1.out -s sig2 -o out2.out  
-compress z
```

Converts: all the signals in the `sig1` file to a `out1.out` file and all the signals in the `sig2` file to a `out2.out.z` file.

# Index

---

## A

absolute voltage 281, 304  
alias 248, 309, 312, 315, 324, 336, 341, 349

## B

back-annotation  
  commands 9  
  controlling connectivity error responses 13, 17, 18, 21, 24, 29, 32, 35  
  specifying a file for 40  
  terminal name mapping 56, 58  
break points  
  listing 254  
  removing 253  
  setting 284

## C

case-sensitivity 1  
commands  
  alias 248, 309, 312, 315, 324, 336, 341, 349  
  categories 8  
  common definitions 4  
  diagnostic 8  
  enable\_ba\_error\_net 13, 17, 18, 21, 24, 29, 32, 35  
  enable\_print\_statement 38  
  enable\_value 4  
  icheck\_node\_zstate 248  
  iclose\_log 251  
  icontinue\_sim 252  
  idelete\_break\_point 253  
  iforce\_node\_voltage 254  
  ilist\_break\_point 254  
  ilist\_force\_node 255  
  imatch\_elem 256  
  imatch\_node 256  
  including in netlist 2  
  interactive mode 2  
    with Tcl 3  
  iopen\_log 258  
  iprint\_connectivity 259

iprint\_dcpath 260  
iprint\_elem\_info 262  
iprint\_exi 264  
iprint\_flash\_cell 266  
iprint\_help 267  
iprint\_node\_info 268  
iprint\_subckt 268  
iprint\_time 269  
iprint\_tree 269  
iprobe\_waveform\_current 270  
iprobe\_waveform\_voltage 272  
iquit\_sim 275  
irelease\_node\_voltage 275  
ireport\_node\_cap 276  
ireport\_operating\_point 280  
isearch\_node 281  
iset\_break\_point 284  
iset\_diagnostic\_option 284  
iset\_env 285  
iset\_interactive\_option 286, 287  
iset\_save\_state 288  
iset\_zstate\_option 289  
load\_ba\_file 40  
load\_vector\_file 55  
map\_ba\_terminal 56, 58  
netlist control 8  
output control 9  
post-layout and back-annotation 9  
probe\_waveform\_current 59  
probe\_waveform\_logic 70  
probe\_waveform\_voltage 74  
report\_power 93  
set\_ccap\_level 125, 127, 130, 133, 144, 149, 150, 152, 202  
set\_dc\_option 133  
set\_duplicate\_rule 137, 140  
set\_floating\_node 141, 143, 154  
set\_meas\_format 155, 156, 175, 177, 179  
set\_message\_option 156  
set\_sample\_point 208, 212  
set\_sim\_level 214, 219, 221  
set\_va\_view 114, 115, 124, 223

## Feedback

### Index

#### D

set\_wildcard\_rule 238  
speed control 9  
use\_mos\_instpar 188, 190  
using a script file 2  
using Tcl mode 3  
vector stimulus 9  
complexity vs. speed trade-off 214, 219, 221  
connectivity errors, controlling 13, 17, 18, 21, 24, 29, 32, 35  
connectivity information 259, 297  
coupling capacitor tolerance, setting 125, 127, 130, 133, 144, 149, 150, 152, 202  
customer support xi

#### D

DC commands  
exit 292  
iclose\_log 293  
icontinue\_dc 293  
idelete\_node\_ic 293  
imatch\_elem 294  
imatch\_node 295  
iopen\_log 296  
iprint\_connectivity 297  
iprint\_elem\_info 298  
iprint\_exi 301  
iprint\_help 302  
iprint\_node\_info 303  
isearch\_node 304  
iset\_node\_ic 306  
debugging mode 2  
diagnostic commands 8  
dynamic CCK commands  
cck\_analog\_pdown 349  
cck\_excess\_ipath 309  
cck\_post 312  
cck\_resistive\_path\_report 354  
cck\_signal 315  
cck\_soa 324  
cck\_toggle\_count 341

#### E

Eldo  
.option statement 2  
print statement 38  
element information 262, 298

enable\_ba\_error\_net 13, 17, 18, 21, 24, 29, 32, 35  
enable\_print\_statement 38  
enable\_value, possible values for 4  
enhanced look-up table 188, 190  
exit 292

#### H

help for interactive commands 267, 302  
hierarchical instance tree 269  
HSPICE  
.option statement 2  
print statement 38  
vector stimulus file 55

#### I

icheck\_node\_zstate 248  
iclose\_log 251, 293  
icontinue\_dc 293  
icontinue\_sim 252  
idelete\_break\_point 253  
idelete\_node\_ic 293  
iforce\_node\_voltage 254  
ilist\_break\_point 254  
ilist\_force\_node 255  
imatch\_elem 256, 294  
imatch\_node 256, 295  
inst\_name 5  
instance definition, using Verilog-A 114, 115, 124, 223  
instance tree 269  
instance\_spec values 4  
-inter flag 2  
interactive commands  
creating aliases for 248, 309, 312, 315, 324, 336, 341, 349  
help 267, 302  
interactive debugging 2  
interactive mode 2  
closing the log file 251, 293  
log file 258, 296  
with Tcl 3  
iopen\_log 258, 296  
iprint\_connectivity 259, 297  
iprint\_dcpth 260

iprint\_elem\_info 262, 298  
iprint\_exi 264, 301  
iprint\_flash\_cell 266  
iprint\_help 267, 302  
iprint\_node\_info 268, 303  
iprint\_subckt 268  
iprint\_time 269  
iprint\_tree 269  
iprobe\_waveform\_current 270  
iprobe\_waveform\_voltage 272  
iquit\_sim 275  
irelease\_node\_voltage 275  
ireport\_node\_cap 276  
ireport\_operating\_point 280  
isearch\_node 281, 304  
iset\_break\_point 284  
iset\_diagnostic\_option 284  
iset\_env 285  
iset\_interactive\_option 286, 287  
iset\_node\_ic 306  
iset\_save\_state 288  
iset\_zstate\_option 289

**L**

load\_ba\_file 40, 49, 54  
load\_vector\_file 55  
LOD models, handling 188, 190  
log file 258, 296  
    controlling the number of messages 156  
look-up table methods 188, 190

**M**

map\_ba\_terminal 56, 58  
.measure file output format 155, 156, 175, 177, 179  
measurements, precise sampling points 208, 212  
models, multiple definitions 137  
module definition, using Verilog-A 114, 115, 124, 223  
MOFSET-specific information 263, 299

**N**

netlist control commands 8  
netlist, including commands in 2

node connectivity information 259, 297  
node information 268, 303

**O**

.option statement 2  
options statement 2  
output control commands 9  
output file format 155, 156, 175, 177, 179

**P**

partitioning instance/module definitions 114, 115, 124, 223  
periodic measurements 208, 212  
post-layout commands 9  
post-layout, specifying a back-annotation file 40  
power consumption 93  
print statements in Eldo or HSPICE netlists 38  
probe\_waveform\_current 59  
probe\_waveform\_logic 70  
probe\_waveform\_voltage 74

**R**

report\_power 93  
reporting DC results 133

**S**

sampling point 208, 212  
script file, issuing commands from 2  
set\_ccap\_level 125, 127, 130, 133, 144, 149, 150, 152, 202  
set\_dc\_option 133  
set\_duplicate\_rule 137, 140  
set\_floating\_node 141, 143, 154  
set\_meas\_format 155, 156, 175, 177, 179  
set\_message\_option 156  
set\_sample\_point 208, 212  
set\_sim\_level 214, 219, 221  
set\_va\_view 114, 115, 124, 223  
set\_wildcard\_rule 238  
simulation  
    continuing a transient simulation 252, 293  
    current time 269  
    level of 214, 219, 221  
    pausing 284

## Feedback

### Index

#### T

terminating 275  
Spectre  
    options statement 2  
speed control commands 9  
speed vs. complexity trade-off 214, 219, 221  
STI models, handling 188, 190  
stimulus file 55  
stop points  
    listing 254  
    removing 253  
stress effects, handling 188, 190  
subcircuit definition, using Verilog-A 114, 115, 124, 223  
subcircuit port names, multiple names 137  
subcircuits, multiple definitions 137  
subckt\_name 5  
support xi

#### T

Tcl mode 3  
    interactive mode with 3  
terminal name mapping 56, 58  
tolerances, setting for coupling capacitors 125, 127, 130, 133, 144, 149, 150, 152, 202

transient simulation, continuing 252, 293

#### U

use\_mos\_instpar 188, 190

#### V

VCD stimulus file 55  
vector stimulus commands 9  
vector stimulus file 55  
Verilog A  
    switching between SPICE and Verilog-A definitions 114, 115, 124, 223

#### W

warning messages 156  
waveform  
    generating node logic values 70  
    probing current 270  
    probing current through instance pins 59  
    probing voltage 74, 272  
wildcard rules 238  
WPE models, handling 188, 190