

# **SpyGlass<sup>®</sup> Connectivity Verify Rules Reference Guide**

---

**Version N-2017.12-SP2, June 2018**



## **Copyright Notice and Proprietary Information**

©2018 Synopsys, Inc. All rights reserved. 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 <https://www.synopsys.com/company/legal/trademarks-brands.html>. 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)

## **Report an Error**

The SpyGlass Technical Publications team welcomes your feedback and suggestions on this publication. Please provide specific feedback and, if possible, attach a snapshot. Send your feedback to [spyglass\\_support@synopsys.com](mailto:spyglass_support@synopsys.com).



# Contents

---

<b>Preface</b> .....	<b>7</b>
<b>About This Book</b> .....	<b>7</b>
<b>Contents of This Book</b> .....	<b>8</b>
<b>Typographical Conventions</b> .....	<b>9</b>
<b>Using the Rules in the SpyGlass Connectivity Verify Product</b> .....	<b>11</b>
<b>Overview</b> .....	<b>12</b>
Features of SpyGlass Connectivity Verify product .....	12
Types of Connectivity Verification Checks .....	13
Require Value Checks .....	13
Connectivity Checks .....	14
Conditional Connectivity Checks .....	15
<b>Licensing Requirements</b> .....	<b>17</b>
<b>Goals in the SpyGlass Connectivity Verify Product</b> .....	<b>18</b>
<b>SpyGlass Connectivity Verify Rule Parameters</b> .....	<b>19</b>
dftAllowNonXValueAtStartOfSensitizedPathInSoc_02.....	20
dft_allow_path_from_enable_to_cgc_clkout .....	20
dft_conn_check_allow_non_x_value_on_sensitizable_path .....	20
dft_conn_check_allow_trace .....	21
dft_conn_check_handle_rtl_negedge.....	21
dft_infer_clock_gating_cell .....	22
dft_max_files_in_a_directory .....	23
dft_require_path_fail_limit.....	23
dft_require_path_invalid_limit .....	24
dft_require_path_pass_limit.....	24
dft_require_path_stop_check_on_pass_count.....	24
dft_soc_unstable_value_sources.....	25
dftShowForcedValues .....	25
dftShowWaveForm.....	26
dft_treat_latches_with_X_on_enable_as_combinational_for_soc_path_checks	
27	
dftUseOffStateOfClockInClockPropagation .....	27
showPowerGroundValue .....	28
<b>Reports in SpyGlass Connectivity Verify Product</b> .....	<b>30</b>

dft_connectivity_check_summary .....	31
--------------------------------------	----

## **Rules in SpyGlass Connectivity Verify ..... 35**

<b>Overview .....</b>	<b>35</b>
<b>Soc_01</b> : Ensure that the expected node value is achieved .....	37
<b>Soc_02</b> : Ensure that the paths between user-specified nodes exist .....	42
<b>Soc_04</b> : Show system state for a given tag. ....	52
<b>Soc_07</b> : Checks the structure between the user-specified nodes .....	55
<b>Soc_08</b> : Checks the path between the user-specified nodes.....	61
<b>Soc_09</b> : Path between user-specified nodes should not exist .....	68
<b>Soc_01_Info</b> : Displays information for node whose expected node value is achieved.....	74
<b>Soc_02_Info</b> : Displays information for the connected user-specified nodes. ....	79
<b>Soc_07_Info</b> : Reports the existence of structure between user-specified nodes.....	86
<b>Soc_10</b> : Reports nets with illegal node values.....	90
<b>Soc_11</b> : Node must satisfy the specified constraint message tag expression	99
<b>Soc_12</b> : Node must not have the specified constraint message tag expression .....	104
<b>Soc_14</b> : Ensure that specified nets are having stable values under specified condition .....	109
<b>Atspeed_21</b> : Check required pulse pattern at specified node. ....	111
<b>Info_Atspeed_21</b> : Expected pulse pattern at the specified node achieved.	115
<b>Diagnose_testmode</b> : Display instances that block the testmode propagation. ....	117
<b>Info_testmode</b> : Display testmode simulation results.....	120

## **Appendix:**

<b>SGDC Constraints .....</b>	<b>125</b>
<b>SGDC Concepts .....</b>	<b>125</b>
<b>SpyGlass Design Constraints .....</b>	<b>126</b>

---

# Preface

---

## About This Book

The SpyGlass® Connectivity Rules Reference describes the SpyGlass rules that check the designs for point to point connectivity and required values on signals with enabling conditions and different modes of operation.

# Contents of This Book

The SpyGlass® Connectivity Rules Reference consists of the following sections:

<b>Section</b>	<b>Description</b>
<i>Using the Rules in the SpyGlass Connectivity Verify Product</i>	How to use the rules in the SpyGlass Connectivity Verify product.
<i>Rules in SpyGlass Connectivity Verify</i>	Detailed description of the rules in the SpyGlass Connectivity Verify product.
<i>Appendix: SGDC Constraints</i>	Tabular list of Constraints required for rules in the SpyGlass Connectivity Verify product.



# Typographical Conventions

This document uses the following typographical conventions:

To indicate	Convention Used
Program code	OUT <= IN;
Object names	OUT
Variables representing objects names	<sig-name>
Message	Active low signal name '<sig-name>' must end with _X.
Message location	OUT <= IN;
Reworked example with message removed	OUT_X <= IN;
Important Information	<b>NOTE:</b> This rule...

The following table describes the syntax used in this document:

Syntax	Description
[ ] (Square brackets)	An optional entry
{ } (Curly braces)	An entry that can be specified once or multiple times
(Vertical bar)	A list of choices out of which you can choose one
. . . (Horizontal ellipsis)	Other options that you can specify



---

# Using the Rules in the SpyGlass Connectivity Verify Product

---

The SpyGlass Connectivity Verify product contains a variety of connectivity-related rules. The rules in SpyGlass Connectivity Verify product assist in verifying required connections, logic values, frequencies, and structures.

This guide covers guidelines, parameter, and rule descriptions for using the Connectivity Verify Product effectively. Please refer to the *SpyGlass Explorer User Guide* for details on using the SpyGlass GUI.

This section explains the following topics:

- [Overview](#)
- [Types of Connectivity Verification Checks](#)
- [Licensing Requirements](#)
- [Goals in the SpyGlass Connectivity Verify Product](#)
- [SpyGlass Connectivity Verify Rule Parameters](#)
- [Reports in SpyGlass Connectivity Verify Product](#)

## Overview

Assembling large structures from existing sub-blocks is a fundamental aspect of contemporary Soc designs. This requires large number of connections from top-level blocks to lower-level blocks. This includes connections between control signals, clock signals, and test logic.

The large amount of connections involved poses challenges in connectivity verification.

In such cases, the SpyGlass Connectivity Verify product enables you to verify:

- Connections between sub-blocks as well as connections from upper-level blocks to lower-level blocks
- Confirmation that logic values on particular pins can be achieved with the proper setup of values on connecting pins
- Verification that values defined on the sgdc files for sub-blocks are achieved when sgdc files for the Soc are applied
- Verification that different frequencies can be achieved at required pins

This section explains the following topics:

- [Features of SpyGlass Connectivity Verify product](#)
- [Types of Connectivity Verification Checks](#)

## Features of SpyGlass Connectivity Verify product

The SpyGlass Connectivity Verify product provides following features:

- Easy capture of connectivity-intent across IP/SoC: The product allows you to:
  - use compact & portable constraints
  - verify one-to-one, one-to-many, many-to-one connections
  - Check for illegal conditions
  - Verify conditional connectivity-checks
  - Validate design methodology consistency across blocks and reuse at SoC level
- Static Checks Supplements Simulation Based Verification:

- ❑ Fast performance to quickly find basic connectivity bugs
- ❑ Supports regression use model
- ❑ Violations clearly state the failure root-cause
- ❑ GUI based design analysis

## Types of Connectivity Verification Checks

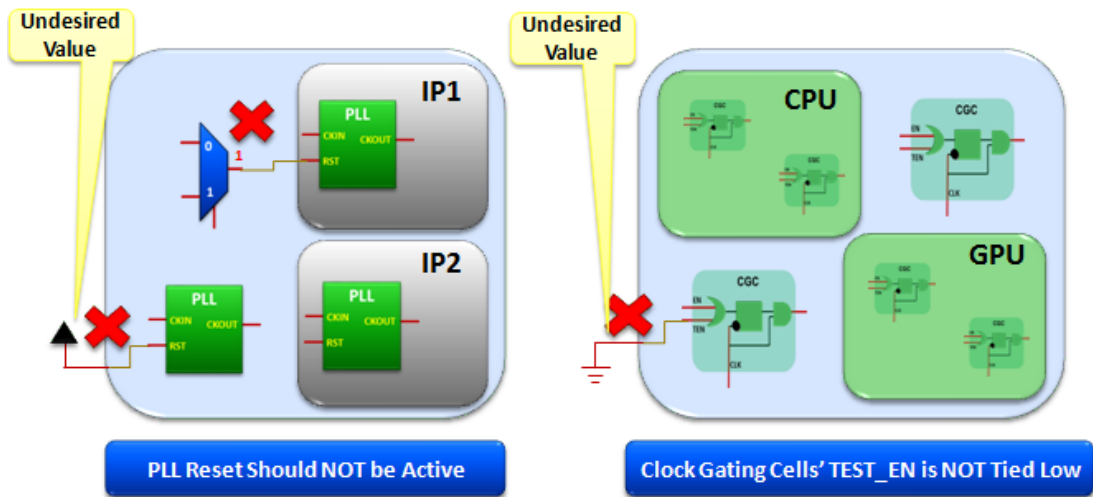
The connectivity verification checks can be classified under the following categories:

- *Require Value Checks*
- *Connectivity Checks*
- *Conditional Connectivity Checks*

### Require Value Checks

The require values checks enables you to verify logic values at different design locations.

*Figure 1* illustrates require value checks under different enabling conditions:



**FIGURE 1.** Require Value Checks

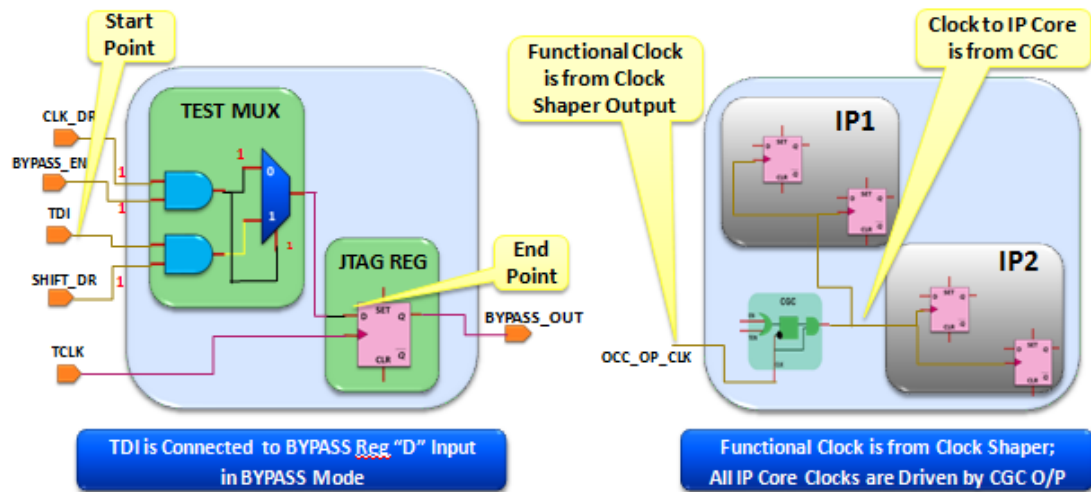
In the above example, the SpyGlass Connectivity Verify product enables you to verify:

- PLL resets are at inactive value
- Clock gating cell (CGC) test enable pins are not tied off

## Connectivity Checks

The connectivity checks enables you to verify that a path exists for specified setup conditions.

*Figure 2* illustrates point to point connectivity checks:



**FIGURE 2.** Connectivity Checks

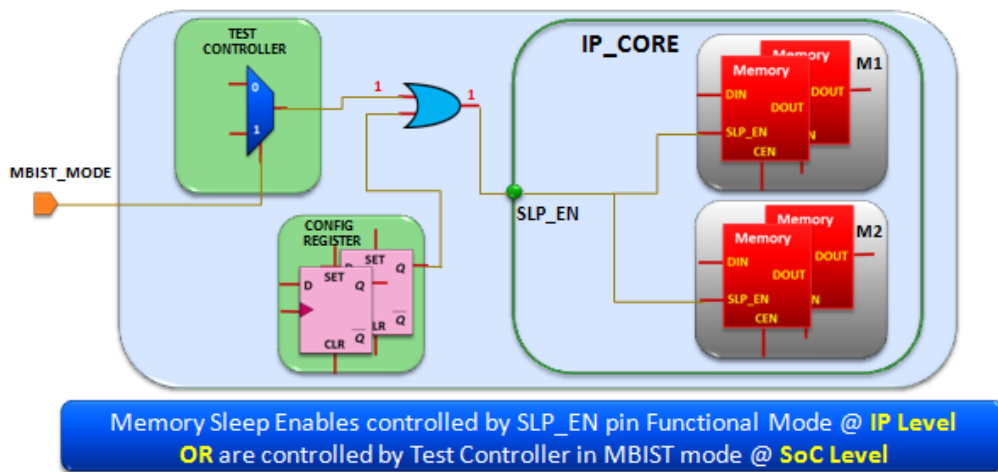
In the above example, the connectivity checks ensure:

- Connectivity from point A to point B across hierarchies
- All Functional clocks are driven from the clock shaper Output
- Core clocks are driven by Clock Gating Cells (CGCs)

## Conditional Connectivity Checks

Conditional connectivity checks enables you to verify complex connections.

*Figure 3* illustrates conditional connectivity checks:



**FIGURE 3.** Conditional Connectivity Checks

In the above example, the conditional checks validate that the memory sleep enables are controlled by:

- SLP\_EN pin at IP-level
- Configuration register in the functional mode at the SoC level



## Licensing Requirements

The SpyGlass Connectivity Verify product requires **soc\_conn\_adv** license.

## Goals in the SpyGlass Connectivity Verify Product

The SpyGlass Connectivity Verify product uses the `connectivity_verification` goal. The `connectivity_verification` goal enables easy capture of connectivity intent across IPs/SoCs and performs static checks to supplement simulation-based verification.

For details, refer to the *GuideWare User Guide*.

## SpyGlass Connectivity Verify Rule Parameters

This section provides explains all the parameters that are used by the rules of the SpyGlass Connectivity Verify product.

You can set these parameters by using the following command in SpyGlass Explorer and Tcl Shell Interface:

```
set_parameter <parameter_name> <parameter_value>
```

For more information on setting the parameters, refer to the SpyGlass *Tcl Interface User Guide* and *SpyGlass Explorer User Guide*.

## dftAllowNonXValueAtStartOfSensitizedPathInSoc\_02

The *dftAllowNonXValueAtStartOfSensitizedPathInSoc\_02* parameter is deprecated.

The non-x nodes are now always allowed as start point for the sensitized [require\\_path](#) check. Therefore, the *dftAllowNonXValueAtStartOfSensitizedPathInSoc\_02* parameter is ignored.

## dft\_allow\_path\_from\_enable\_to\_cgc\_clkout

Use this parameter to allow a connectivity path from enable (data and test) to CGC clock-out pin.

Used by	<a href="#">Soc_02</a> , <a href="#">Soc_02_Info</a> , <a href="#">Soc_08</a> , <a href="#">Soc_09</a>
Options	on, off
Default value	off
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_allow_path_from_enable_to_cgc_clkout on</code>
<i>Usage in goal/source files</i>	<code>-dft_allow_path_from_enable_to_cgc_clkout=on</code>

## dft\_conn\_check\_allow\_non\_x\_value\_on\_sensitizable\_path

Allows non-x value (0 or 1) on a sensitizable path.

By default, the value of the parameter is on. Therefore, the [Soc\\_02](#) and [Soc\\_02\\_Info](#) rules allow non-x value on the path while performing checks on the sensitizable paths.

Set the value of the parameter to off to ignore non-X value on the path while performing checks on the sensitizable paths.

Used by	<a href="#">Soc_02</a> , <a href="#">Soc_02_Info</a>
Options	on, off
Default value	on
<b>Example</b>	

<i>Console/Tcl-based usage</i>	set_parameter dft_conn_check_allow_non_x_value_on_sensitizable_path off
<i>Usage in goal/source files</i>	- dft_conn_check_allow_non_x_value_on_sensitizable_path=off

## dft\_conn\_check\_allow\_trace

Use this parameter to allow combinational traversal through the asynchronous pins of a flip-flop while performing the checks for require path (*require\_path* constraint) or illegal path (*illegal\_path* constraint).

Used By	Conn_02, <a href="#">Soc_02</a> , <a href="#">Soc_02_Info</a> , Conn_08, Conn_09, <a href="#">Soc_08</a> , <a href="#">Soc_09</a>
Options	on, off
Default Value	off
Example	
Console/Tcl-based Usage	set_parameter dft_conn_check_allow_trace_through_async on
Usage in goal/source files	-dft_conn_check_allow_trace_through_async=on

## dft\_conn\_check\_handle\_rtl\_negedge

Considers the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.

By default the value of the parameter is off. In this case, the CP/CLR/PRE pin of the flip-flop remains the start/end point of the connectivity check.

Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.

Used by	<b>DFT:</b> Conn_01, Conn_02, Conn_08, Conn_09, Conn_10 <b>Connectivity Verify:</b> <a href="#">Soc_01</a> , <a href="#">Soc_01_Info</a> , <a href="#">Soc_02</a> , <a href="#">Soc_02_Info</a> , <a href="#">Soc_08</a> , <a href="#">Soc_09</a> , <a href="#">Soc_10</a>
Options	on, off
Default value	off
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_conn_check_handle_rtl_negedge off</code>
<i>Usage in goal/source files</i>	<code>-dft_conn_check_handle_rtl_negedge = off</code>

## dft\_infer\_clock\_gating\_cell

Selects the behavior of automatic clock gating cell (CGC) inference.

By default, the value of this parameter is set to `on` and the cells similar to Clock Gating Cells (CGCs) are treated as CGCs.

Set the value of the parameter to `off` to turn off CGC inference.

See Identifying Clock Gating Cells section for more information on the ways to infer CGCs.

For more information on identifying CGCs, refer to the *Identifying Clock Gating Cells* section of the *SpyGlass DFT Rules Reference Guide*.

Used by	All SpyGlass Connectivity Verify Rules
Options	on, off
Default value	on
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_infer_clock_gating_cell off</code>
<i>Usage in goal/source files</i>	<code>-dft_infer_clock_gating_cell=off</code>

## dft\_max\_files\_in\_a\_directory

Specifies maximum number of csv files in a single directory.

Used by	All SpyGlass Connectivity Verify Rules
Options	<any natural number>
Default value	2000
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_max_files_in_a_directory 5000</code>
<i>Usage in goal/source files</i>	<code>-dft_max_files_in_a_directory=5000</code>

## dft\_require\_path\_fail\_limit

Selects the number of violations reported by the [Soc\\_02](#) and [Soc\\_08](#) rules for the [require\\_path](#) and [require\\_strict\\_path](#) constraints failure when either the `-from_one_of` or `-to_one_of` arguments of the constraints are not specified.

Used by	<a href="#">Soc_02</a> , <a href="#">Soc_08</a>
Options	<any natural number>
Default Value	10
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_require_path_fail_limit -1</code>
<i>Usage in goal/source files</i>	<code>-dft_require_path_fail_limit=-1</code>

## dft\_require\_path\_invalid\_limit

Limits the number of invalid path violations reported by the Soc\_08 rule for the [require\\_strict\\_path](#) constraint failure.

Used by	<a href="#">Soc_08</a>
Options	<any natural number>
Default Value	10
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_require_path_invalid_limit -1</code>
<i>Usage in goal/source files</i>	<code>--dft_require_path_invalid_limit=-1</code>

## dft\_require\_path\_pass\_limit

Limits the number of violations reported by the [Soc\\_02\\_Info](#) and [Soc\\_08](#) rules for the [require\\_path](#) and [require\\_strict\\_path](#) constraints success when either the `-from_one_of` or `-to_one_of` arguments of the constraints are specified.

Used by	<a href="#">Soc_02_Info</a> , <a href="#">Soc_08</a>
Options	<any natural number>
Default Value	-1
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_require_path_pass_limit 1</code>
<i>Usage in goal/source files</i>	<code>-dft_require_path_pass_limit=1</code>

## dft\_require\_path\_stop\_check\_on\_pass\_count

Limits the number of violations reported by the [Soc\\_02\\_Info](#) rule when both the conditions are true:



## SpyGlass Connectivity Verify Rule Parameters

- Either the `-from_one_of` or `-to_one_of` arguments for the `require_path` constraint is specified
- The `require_path` constraint check is successful

Used by	<a href="#">Soc_02_Info</a>
Options	<any natural number>
Default Value	-1
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_require_path_stop_check_on_pass_count 1</code>
<i>Usage in goal/source files</i>	<code>-dft_require_path_stop_check_on_pass_count=1</code>

## dft\_soc\_unstable\_value\_sources

Specifies unstable value sources, other than scannable flip-flops and latches, that needs to be reported by the [Soc\\_14](#) rule.

Used by	<a href="#">Soc_14</a>
Options	<code>none, all, blackbox, hanging_net, port</code>
Default Value	<code>none</code>
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dft_soc_unstable_value_sources none</code>
<i>Usage in goal/source files</i>	<code>-dft_soc_unstable_value_sources=none</code>

## dftShowForcedValues

Use the `dftShowForcedValues` rule parameter to differentiate between the values enforced by the user and values that have been propagated automatically by the rule.

The `dftShowForcedValues` rule parameter controls the display of

signals in the schematic view. When the `dftShowForcedValues` is set, signals forced onto specific nodes by the `test_mode` constraints appear as 0(F) or 1(F). Signals that result or are implied from these forced signals appear as 0 or 1 (without the (F) suffix).

This provides clear differentiation between causal signals and result signals. For example, in a design where the output of an AND gate is set to 1 by a `test_mode` constraint and where other constraints on nodes in the fan-in cone for AND cause a 0 on one of the AND gate inputs, the output of the AND will retain its forced value. Therefore, the inconsistency of a 0 on an AND input and a 1 on the AND output may be resolved.

When the `dftShowForcedValues` is not set, the (F) suffix does not appear.

Used by	<a href="#">Info_testmode</a>
Options	off, on
Default value	on
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dftShowForcedValues off</code>
<i>Usage in goal/source files</i>	<code>-dftShowForcedValues=off</code>

## dftShowWaveForm

Use this parameter to select whether or not to display the waveform information for the [Info\\_testmode](#) rule.

Used by	<a href="#">Info_testmode</a>
Options	on, off
Default value	off
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<code>set_parameter dftShowWaveForm on</code>
<i>Usage in goal/source files</i>	<code>-dftShowWaveForm=on</code>

## dft\_treat\_latches\_with\_X\_on\_enable\_as\_combinational\_for\_soc\_path\_checks

Defines the treatment of latches, that is, whether to consider them as combinational or sequential, where enable pin does not get either 0 or 1, when running Soc path check rules.

Used by	<a href="#">Soc_02</a> , <a href="#">Soc_02_Info</a> , <a href="#">Soc_08</a> , <a href="#">Soc_09</a>
Options	on, off
Default value	off
<b>Example</b>	
<i>Console/Tcl-based usage</i>	<pre>set_parameter dft_treat_latches_with_X_on_enable_as_combinational_for_soc_path_checks on</pre>
<i>Usage in goal/source files</i>	<pre>- dft_treat_latches_with_X_on_enable_as_combinational_for_soc_path_checks = on</pre>

## dftUseOffStateOfClockInClockPropagation

Specifies the treatment of the off state of the clocks during shift, capture, or atspeed mode.

By default, the `dftUseOffStateOfClockInClockPropagation` parameter is set to `on` to simulate the off state of the test clock, that is, 0 for `rtz` and 1 for `rto`. It also keeps functional clocks to unknown state during the shift mode.

Set this parameter to `off` to keep all clocks, including functional and test clocks to unknown state. See the table below to view the list of other possible values.

The following table lists the clock state and the corresponding event applied on it during scanshift mode simulation:

Clock State	Event Applied
Unknown	No event is applied

Clock State	Event Applied
off	0X (rtz) or 1X (rto)
free running	0X1X

An 'X' is always simulated at the end is that clock line is free during clock propagation. You can use this parameter when ICGs are used to drive constant value.

Used by	All rules in the SpyGlass Connectivity Verify product
Options	off, on, fclk_unknown_n_tclk_unknown, fclk_unknown_n_tclk_off_state, fclk_unknown_n_tclk_free_running, fclk_off_state_n_tclk_off_state, fclk_off_state_n_tclk_free_running, fclk_free_running_n_tclk_off_state, fclk_free_running_n_tclk_free_running
Default value	on
<b>Example</b>	
<i>Console/Tcl-based usage</i>	set_parameter dftUseOffStateOfClockInClockPropagation off
<i>Usage in goal/source files</i>	-dftUseOffStateOfClockInClockPropagation= fclk_free_running_n_tclk_off_state

## showPowerGroundValue

Use this parameter to control the schematic display for power/ground simulation.

When the value of the parameter is set to on, the simulation value of a net due to power/ground is displayed. To hide the power/ground simulation values of a net, set this switch to 'off'.

Used by	<a href="#">Info_testmode</a>
Options	on, off
Default value	on
<b>Example</b>	

---

SpyGlass Connectivity Verify Rule Parameters

---

<i>Console/Tcl-based usage</i>	<code>set_parameter showPowerGroundValue off</code>
--------------------------------	---

---

<i>Usage in goal/source files</i>	<code>-showPowerGroundValue=off</code>
-----------------------------------	--

---

## Reports in SpyGlass Connectivity Verify Product

The SpyGlass Connectivity Verify product generates the rules and reports that you can view from the *Reports* menu in SpyGlass Explorer or by using the `set_option report <report-name>` command followed by the report name.

The SpyGlass Connectivity Verify product generates [dft\\_connectivity\\_check\\_summary](#) report that contains a summary of the Soc\_01, Soc\_02, Soc\_07, Soc\_08, Soc\_09, Soc\_01\_Info, Soc\_02\_Info, and Soc\_07\_Info rules.

## dft\_connectivity\_check\_summary

The dft\_connectivity\_check\_summary.rpt file is generated by the [Soc\\_01](#), [Soc\\_02](#), [Soc\\_07](#), [Soc\\_08](#), [Soc\\_09](#), [Soc\\_01\\_Info](#), [Soc\\_02\\_Info](#), and [Soc\\_07\\_Info](#) rules. This report contains a summary of these rules.

```
#####
# Purpose :
#   The report summarizes all the connectivity check results
#
#   require_value:      Checked by Soc_01, Soc_01_Info
#   Reports the number of 'require_value' constraints
#   passed and failed
#
#   require_path:      Checked by Soc_02, Soc_02_Info
#   Reports the number of 'require_path' constraints passed
#   and failed
#
#   require_structure: Checked by Soc_07, Soc_07_Info
#   Reports the number of 'require_structure' constraints
#   passed and failed
#
#   require_strict_path: Checked by Soc_08
#   Reports the number of 'require_strict_path' constraints
#   passed and failed
#
#   illegal_path:      Checked by Soc_09
#   Reports the number of 'illegal_path' constraints passed
#   and failed
#
#   NOTE : For more details, refer to the corresponding
#   rule violations
#####

#####

# Format :
#   Top design unit: <top_design_name>
#
```

```
#      require_value:
#      <N> require_value constraints passed
#      <N> require_value constraints failed
#
#      require_path:
#      <N> require_path constraints passed
#      <N> require_path constraints failed
#
#      require_structure:
#      <N> require_structure constraints passed
#      <N> require_structure constraints failed
#
#      require_strict_path:
#      <N> require_strict_path constraints passed
#      <N> require_strict_path constraints failed
#
#      illegal_path:
#      <N> illegal_path constraints passed
#      <N> illegal_path constraints failed
#
#      NOTE : If a particular rule is disabled in the current
#      run, then there will not be a section corresponding to
#      that rule
#####
#####

Top design unit: top1

require_value:
  '1' require_value constraint passed
  No require_value constraint failed

require_path:
  '2' require_path constraints passed
  No require_path constraint failed

require_structure:
```



```
No require_structure constraint passed
'1' require_structure constraint failed
```

```
require_strict_path:
'2' require_strict_path constraints passed
'2' require_strict_path constraints failed
```

```
illegal_path:
'1' illegal_path constraint passed
'2' illegal_path constraints failed
```

```
#####
```

```
#####
```

```
Top design unit: top2
```

```
require_value:
No valid require_value constraint specified
```

```
require_path:
No valid require_path constraint specified
```

```
require_structure:
No valid require_structure constraint specified
```

```
require_strict_path:
No valid require_strict_path constraint specified
```

```
illegal_path:
No valid illegal_path constraint specified
```

```
#####
```



---

# Rules in SpyGlass Connectivity Verify

---

## Overview

The SpyGlass<sup>®</sup> Connectivity Verify product has the following rules:

<b>Rule</b>	<b>Flags/Highlights</b>
<i>Soc_01</i>	Nodes that do not achieve expected values
<i>Soc_02</i>	Missing net connections between specified nodes
<i>Soc_04</i>	Simulation results for the conditions specified by a tag
<i>Soc_07</i>	Structures that are not same as the user-specified structures between the user-specified nodes
<i>Soc_08</i>	The path between the user-specified nodes
<i>Soc_09</i>	Path between user specified nodes should not exist
<i>Soc_01_Info</i>	Nodes that achieve expected values
<i>Soc_02_Info</i>	Valid net connections between specified nodes
<i>Soc_07_Info</i>	Structures that are same as the user-specified structures between the user-specified nodes
<i>Soc_10</i>	Reports nets with illegal node values
<i>Soc_11</i>	Ensures that the node satisfies the specified constraint message tag expression

<b>Rule</b>	<b>Flags/Highlights</b>
<i>Soc_12</i>	Ensures that the node does not have the specified constraint message tag expression
<i>Soc_14</i>	Ensure that specified nets are having stable values under specified condition
<i>Atspeed_21</i>	Check required pulse pattern at specified node
<i>Info_Atspeed_21</i>	Expected pulse pattern at the specified node achieved.
<i>Diagnose_testmode</i>	Display instances that block the testmode propagation.
<i>Info_testmode</i>	Display testmode simulation results

## Soc\_01

**Ensure that the expected node value is achieved**

### When to Use

Use this rule to identify the nodes that do not achieve the expected simulation value.

### Description

The Soc\_01 rule generates the SpyGlass Explorer highlight data for those nodes specified with *require\_value* constraints that do not achieve the specified simulation value when the specified tag condition is simulated.

Expected values at arbitrary nodes and the applied values that should cause or force the expected values are checked. This is useful for ensuring that unit-level test requirements are satisfied at the SoC level.

Consider the following *require\_value* constraint:

```
require_value
  -tag <tagName> -name <nodeNames> -value <value>
```

The Soc\_01 rule generates the SpyGlass Explorer highlight data for those nodes specified with the *-name* argument that do not achieve the simulation value *<value>* when the conditions of *<tagName>* are simulated.

To view the nodes that achieve the expected value, use the [Soc\\_01\\_Info](#) rule.

### Prerequisites

Specify the *require\_value* constraint.

### Default Weight

10

### Language

Verilog, VHDL

### Method

For each *define\_tag*, simulate power, ground and all conditions defined for this *define\_tag*.

For each *require\_value* with the current *define\_tag*, check that the defined pin has the

defined value. Otherwise, report a message.

## Parameter(s)

- *dft\_conn\_check\_handle\_rtl\_negedge*: Default value is off. Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.
- *dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

## Constraint(s)

- *define\_tag*: Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.
- *require\_value*: Use this constraint to define a check that requires a logic value to be established on a certain node when the circuit has been simulated using the condition specified by the -tag argument.

## Operating Mode

Define\_tag

## Messages and Suggested Fix

The following violation messages are displayed for the Soc\_01 rule:

### Message 1

[ERROR] [constraint\_message\_tag: <value>] Node <name> has value <value1>. (Required: <value2>) under tag <tag-name>

### Arguments

- Constraint tag value, <value>
- Name of the node <name>
- Actual value <value1>
- Expected value <value2>
- Tag name <tag-name>

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the -constraint\_message\_tag argument for the [require\\_value](#) constraint.*

**Potential Issues**

A violation is reported due to incomplete or incorrect simulation condition or incorrect design connectivity.

**Consequences of Not Fixing**

Not fixing the violation may result in unexpected code behavior.

**How to Debug and Fix**

View the Incremental Schematic for the violation message. The Incremental Schematic highlights the node where the required simulation mismatches the actual simulation.

You can also view the violation for the Soc\_04 rule along with the violation of the Soc\_01 rule in the Incremental Schematic window. To do this, double-click the violation for the Soc\_01 rule and open the Incremental Schematic window.

The violation message for the Soc\_04 rule overlaps the violation message for the Soc\_01 rule in the Incremental Schematic window. This is useful in debugging the violation for the Soc\_01 rule.

To fix the violation, see Example Code and/or Schematic section.

**Message 2**

**[WARNING]** [constraint\_message\_tag: <value>] illegal\_value command is using option -matchNBits (<value1>) without setting parameter dftShowWaveForm to 'on' under <tag-name>. Setting -matchNBits to value '1' for the rule checking purpose

**Arguments**

- Constraint tag value, <value>
- Illegal value <value1>
- Tag name <tag-name>

**Potential Issues**

A violation is reported when the following conditions hold true:

- Value of the *dftShowWaveForm* parameter is not set on
- The *require\_value* constraint uses the -matchNbits <value> argument under <tag-name> condition
- The value of the -matchNBits argument is greater than 1

The Soc\_01 rule does not report this violation message if the tag is defined using the [define\\_tag](#) command.

### ***Consequences of Not Fixing***

Not fixing the violation may result in unexpected code behavior.

### ***How to Debug and Fix***

To fix the violation, set the value of the [dftShowWaveForm](#) parameter to on.

## **Example Code and/or Schematic**

Consider the following example:

```
require_value -name mode2 -value 0 -matchNBits 1 -useTestmode  
require_value -name mode3 -value 1 -matchNBits 1 -useTestmode
```



## Overview

The screenshot displays the Atrenta Console interface. The top window, titled "Incremental Schematic : tap", shows a circuit diagram for the "INSTR\_DEC" block. It includes components like "rtl\_c\_118", "RTL\_MUX2", "mode0\_reg", "RTL\_FDC", "rtl\_c\_116", and "RTL\_MUX2". The schematic shows signal paths and values for "mode1" (1) and "mode0" (0(1)).

The bottom window, titled "View: Message", shows the results of a test run. The messages are as follows:

```

ERROR: [ 1 ]
- Soc_01[1]:Expected node value must be achieved.
  Node 'mode0' has value 0 (Required: 1) under tag '-useTestmode', tap.sgdc, 25
INFO: [ 5 ]
- Soc_01_Info[3]:Expected node value is achieved.
  Node 'mode1' has required value(1) under tag '-useTestmode', tap.sgdc, 26
  Node 'mode2' has required value(0) under tag '-useTestmode', tap.sgdc, 27
  
```

**Default Severity Label**

Error

**Rule Group**

SoC

**Reports and Related Files**

[dft\\_connectivity\\_check\\_summary.rpt](#): Reports the number of *require\_value* constraints passed and failed.

## Soc\_02

**Ensure that the paths between user-specified nodes exist**

### When to Use

Use this rule to identify the disjointed pair of user-specified nodes.

### Rule Description

The Soc\_02 rule reports connection violations between specified pair of nodes.

The Soc\_02 rule checks either conditional or unconditional paths between user-specified nodes.

**NOTE:** *The Soc\_02 rule checks all [require\\_path](#) constraints — without the `-tag` argument and with the `-tag` argument (earlier checked by the now obsolete Soc\_03 rule.)*

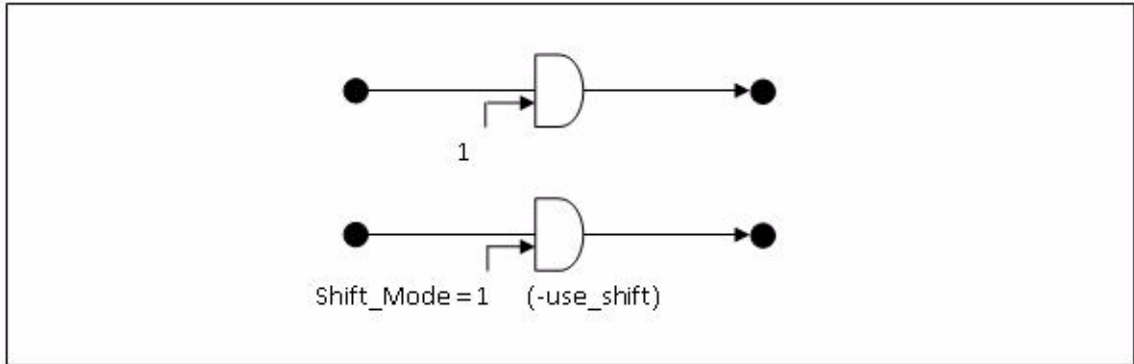
While processing the [require\\_path](#) constraints, Soc\_02 rule checking depends on `-path_type` and simulation condition as discussed below:

Specified field of <code>require_path</code> constraint	Simulation condition
<code>-use_shift</code>	Shift
<code>-use_capture</code>	Capture
<code>-use_captureATspeed</code>	Capture (atspeed)
<code>-tag &lt;tag_name&gt;</code>	tag_name
If none of the above specified	Power ground

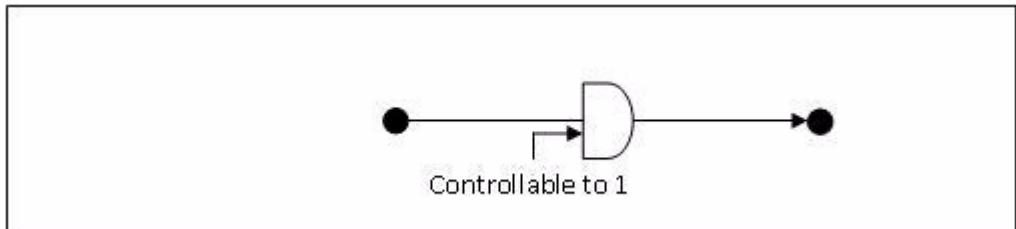
- If you specify the value of the `-path_type` argument as `sensitized`, the Soc\_02 rule performs the strict functional checking. This ensures that the path is properly sensitized by the specified simulation condition.
- If you specify the value of the `-path_type` argument as `sensitizable`, the Soc\_02 rule performs the functional checking. This ensures that the path is properly sensitizable and is not blocked by the simulation condition.
  - If you specify the value of the `-path_type` argument as `buffered`, the Soc\_02 rule looks for strict topological checking and checks for buffers and inverters only.

- ❑ If you do not specify the value of the `-path_type` argument, the default value of the argument, that is, `sensitizable` is used.

Consider the following example, where `Shift_Mode` is equal to 1, that is, `-use_shift` parameter is specified.



The above figure shows an example of the sensitized path. Here, the value of the `-path_type` argument is set as `sensitized`, which ensures the existing logic connectivity between the start and end points. Consider the following example, where the clock is controllable to 1:



The above figure shows an example of the sensitizable path. Here, the value of the `-path_type` argument is specified as `sensitizable`, which checks whether the target path is sensitizable during the specific simulation condition.

SoC integration often requires that connections between various units exist. This rule allows arbitrary from-to pin to be checked.

To view the valid paths, use the [Soc\\_02\\_Info](#) rule.

The `Soc_02` rule flags a violation if during traversal to find a path between

two nodes, a contentious net is found. This path is no longer considered a correct path, therefore, the [Soc\\_02\\_Info](#) rule does not highlight this path.

**NOTE:** *If you do not specify the `-undirected` qualifier in the [require\\_path](#) constraint, the `Soc_02` rule traces the path from `-from` node to `-to` node only.*

### Prerequisites

Specify the [require\\_path](#) constraint.

### Default Weight

10

### Language

Verilog, VHDL

### Method

If specified, simulate require path for `<path_type>` verification under specified simulation condition.

For each from-to pair of `require_path` specification:

Walk the unblocked fan-out cone under the simulation condition from the start point while maintaining the phase inversion. If the walk doesn't terminate on the specified endpoint for the required `<path_type>`, report a violation.

## Parameter(s)

- [dftAllowNonXValueAtStartOfSensitizedPathInSoc\\_02](#): This parameter is deprecated and is ignored for rule-checking.
- [dft\\_allow\\_path\\_from\\_enable\\_to\\_cgc\\_clkout](#): The default value is off. Set the value of the parameter to on to allow a connectivity path from enable (data and test) to CGC clock-out pin.
- [dft\\_conn\\_check\\_allow\\_non\\_x\\_value\\_on\\_sensitizable\\_path](#): The default value is on. Set the value of the parameter to off to ignore non-X (0 or 1) value on the path while performing checks on the sensitizable paths.
- [dft\\_conn\\_check\\_handle\\_rtl\\_negedge](#): Default value is off. Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.
- [dft\\_require\\_path\\_fail\\_limit](#): The default value is 10. Set the value of the parameter to any natural number to control the number of violations reported by the `Soc_02` rule for the `require_path` constraint failure when

either the `-from_one_of` or `-to_one_of` arguments of the `require_path` constraint is specified.

- `dft_treat_latches_with_X_on_enable_as_combinational_for_soc_path_checks`: The default value is off. Set the value of the parameter to on to define the treatment of latches. That is, whether to consider them as combinational or sequential, where enable pin does not get either 0 or 1, when running Soc path check rules.
- `dftUseOffStateOfClockInClockPropagation`: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.
- `dft_conn_check_allow_trace`: The default value is off. Set the value of the parameter to on to allow combinational traversal through the asynchronous pins of a flip-flop while performing the checks for `require_path` ( `require_path` constraint) or illegal path ( `illegal_path` constraint).

## Constraint(s)

- `require_path` (mandatory): Use this constraint to define a connectivity check for a path from a pin specified with the `-from` argument to a pin specified with the `-to` argument.
- `test_mode` (optional): Use this constraint to specify the set of conditions, both pins and values, that when simulated, will force the circuit in test mode.
- `define_tag` (optional): Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.

## Operating Mode

Scanshift, Capture, Power Ground, Define\_tag

## Messages and Suggested Fix

### Message 1

```
[ERROR] [constraint_message_tag: <value>] '<path_type>' path(s)
not found from '<from_node_name>' to '<to_node_count>' nodes
under <simulation condition> within '<sequential_depth>'
sequential depth
```

### Arguments

To view the list of message arguments, click [Arguments](#).

### **Potential Issues**

The violation message appears if no valid path exists between the user-specified nodes.

### **Consequences of Not Fixing**

Not fixing the violation may result in unsearchable expected path. This may impact the functionality of the design.

### **How to Debug and Fix**

For more information on debugging and fixing the violation, click [How to Debug and Fix](#).

### **Message 2**

**[ERROR]** [constraint\_message\_tag: <value>] '<path\_type>' path(s) not found to '<to\_node\_name>' from '<from\_node\_count>' nodes under <simulation condition> within '<sequential\_depth>' sequential depth

### **Arguments**

To view the list of message arguments, click Arguments.

### **Potential Issues**

The violation message appears if no valid path exists between the user-specified nodes.

**NOTE:** *This violation message is reported instead of Message 1, in case of '-from\_one\_of' field specified in 'require\_path' constraint.*

### **Consequences of Not Fixing**

Not fixing the violation may result in unsearchable expected path. This may impact the functionality of the design.

### **How to Debug and Fix**

For more information on debugging and fixing the violation, click [How to Debug and Fix](#).

### **Message 3**

**[ERROR]** [constraint\_message\_tag: <value>] '<path\_type>' path(s) not found from '<from\_node\_name>' to '<to\_node\_count>' nodes within '<sequential\_depth>' sequential depth

### **Arguments**

To view the list of message arguments, click [Arguments](#).

### **Potential Issues**

The violation message appears if no valid path exists between the user-specified nodes.

**NOTE:** *This violation message is reported instead of Message 1, when '-path\_type' is 'buffered' in 'require\_path' constraint.*

### **Consequences of Not Fixing**

Not fixing the violation may result in unsearchable expected path. This may impact the functionality of the design.

### **How to Debug and Fix**

For more information on debugging and fixing the violation, click [How to Debug and Fix](#).

### **Message 4**

**[ERROR]** [constraint\_message\_tag: <value>] '<path\_type>' path(s) not found to '<to\_node\_name>' from '<from\_node\_count>' nodes within '<sequential\_depth>' sequential depth

### **Arguments**

- Constraint tag value, <value>
- Expected path type, as specified by 'require\_path' constraint, <path\_type>
- Node name specified in the '-from' field of 'require\_path' constraint, <from\_node\_name>
- Node name specified in the '-to' field of 'require\_path' constraint, <to\_node\_name>
- No of '-to' nodes for which connectivity check failed from the specified '-from' node, <to\_node\_count>
- No of '-from' nodes from which connectivity check failed to the specified '-to' node, <from\_node\_count>
- Value specified by '-sequential\_depth' field of 'require\_path' constraint, <sequential\_depth>
- Simulation condition specified in the 'require\_path' constraint, <simulation\_condition>

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the*

-*constraint\_message\_tag* argument for the [require\\_path](#) constraint.

### **Potential Issues**

The violation message appears if no valid path exists between the user-specified nodes.

**NOTE:** *This violation message is reported instead of Message 2, when '-path\_type' is 'buffered' in 'require\_path' constraint.*

### **Consequences of Not Fixing**

Not fixing the violation may result in unsearchable expected path. This may impact the functionality of the design.

### **How to Debug and Fix**

Double-click the violation message to open the spreadsheet, which contains the list of all the violation messages. Click on a violation message in the spreadsheet to view the incremental schematic of the violation message. The incremental schematic displays the path specified in the violation message.

If endpoints are displayed start probing from either fan-out cone of from point or fan-in cone of to point to see why required path does not exist.

If the path from start point up to the instance which is blocking is displayed then analyze the text annotation date displayed to see why the path is blocked at the instance highlighted.

To fix the violation, see [Example Code and/or Schematic](#).

## **Message 5**

```
[ERROR] [constraint_message_tag: <value>] ' <no_of_paths>'
(<limit_breached>) ' <path_type>' path(s) found from
' <from_node_name> node under <simulation condition> within ' <
sequential_depth >' sequential depth
```

### **Arguments**

- Constraint tag value, <value>
- Expected path type, as specified by 'require\_path' constraint, <path\_type>
- Limit to the number of paths, <limit\_breached>
- Node name specified in the '-from' field of 'require\_path' constraint, <from\_node\_name>



- Simulation condition specified in the 'require\_path' constraint, <simulation\_condition>
- Value specified by '-sequential\_depth' field of 'require\_path' constraint, <sequential\_depth>

### **Potential Issues**

Specified number of paths are not present between user-specified nodes.

### **Consequences of Not Fixing**

Not fixing the violation may result in unsearchable expected path. This may impact the functionality of the design.

### **How to Debug and Fix**

Double-click the violation message to open the spreadsheet, which contains the list of all the violation messages. Click on a violation message in the spreadsheet to view the incremental schematic of the violation message. The incremental schematic displays the path specified in the violation message.

If endpoints are displayed start probing from either fan-out cone of from point or fan-in cone of to point to see why required path does not exist.

If the path from start point up to the instance which is blocking is displayed then analyze the text annotation date displayed to see why the path is blocked at the instance highlighted.

To fix the violation, see [Example Code and/or Schematic](#).

### **Message 6**

**[WARNING]** [constraint\_message\_tag: <value>]  
 min\_to\_paths(' <min\_path\_value>') should have a value less than  
 max\_to\_paths(' <max\_path\_value>'), ignoring these values

### **Arguments**

- Constraint tag value, <value>
- Minimum number of expected successful paths, <min\_path\_value>
- Maximum number of expected successful paths, <max\_path\_value>

### **Potential issues**

The value specified is ignored.

### **How to debug and fix**

Double-click the violation message to open the spreadsheet, which contains the list of all the violation messages. Click on a violation message in the spreadsheet to view the incremental schematic of the violation message. The incremental schematic displays the path specified in the violation message.

If endpoints are displayed start probing from either fan-out cone of from point or fan-in cone of to point to see why required path does not exist.

If the path from start point up to the instance which is blocking is displayed then analyze the text annotation date displayed to see why the path is blocked at the instance highlighted.

To fix the violation, review the constraint and modify to specify valid values.

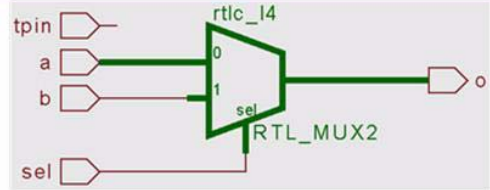
### Example Code and/or Schematic

Consider the following example:

■ Note: in both the scenarios the select pin of the mux is 'unconstrained'

**Case 1**

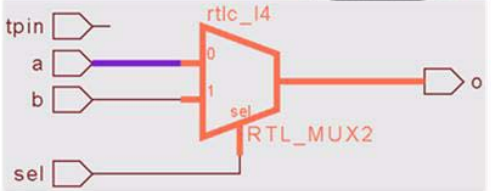
```
define_tag-tag FUNC_mode-name tpin-value 1
require_path -from a-to o -tag FUNC_mode
-path_type sensitized
```



Soc\_02\_Info[1]:Connection between user specified nodes exists.

Path from point 'a' to point 'o' is 'sensitized' under power ground simulation condition, test.sgdc\_3

```
define_tag-tag FUNC_mode-name tpin-value 1
require_path -from a-to o -tag FUNC_mode
-path_type sensitized
```



Soc\_02[1]:Paths between user specified nodes must exist.

Path from point 'a' to 'o' is 'not-sensitized' at node 'test.o' under power ground simulation condition, test.sgdc\_3

**Case 2**

### Schematic highlight

Start and end point would be highlighted. (No topological connection)

Path from start point till the place signal reached, rttc would be highlighted in

one color and from there to end point in different color. (Where path is blocked)

Path with the same phase in one color and invert phase in different color highlighted. This schematic is applicable only when the `-invert` or `-noinvert` argument is specified in the [require\\_path](#) constraint.

## Default Severity

### Label

Error

## Rule Group

SoC

## Reports and Related Files

[dft\\_connectivity\\_check\\_summary](#).rpt: Reports the number of [require\\_path](#) constraints passed and failed.

## Soc\_04

**Show system state for a given tag.**

### When to Use

Use this rule to show the system state for each condition specified by the `define_tag` constraint.

### Description

The Soc\_04 rule shows the simulation results for each condition specified by the `define_tag` constraint.

The state of user-selected nodes, when a set of nodes are in a particular state, is checked. This is useful to verify that port requirements on one or more blocks in a design have required values when a set of nodes have particular values.

### Prerequisites

Specify the `define_tag` constraint.

### Default Weight

10

### Language

Verilog, VHDL

### Method

For each set of `define_tag` conditions

Simulate power, ground and the defined node/value pairs

### Parameter(s)

- `dftShowWaveForm` (optional): The default value is off. Set the value of the parameter to on to enable the generation of waveform corresponding to the rule message in the Waveform viewer of the SpyGlass Explorer.
- `dftUseOffStateOfClockInClockPropagation`: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

## Constraint(s)

[define\\_tag](#) (optional): Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.

## Operating Mode

Define\_tag

## Messages and Suggested Fix

### Message 1

[INFO] Tag '<tag-name>' is displayed for design '<du-name>'

### Arguments

To view list of arguments, click [Arguments](#).

### Potential Issues

Since this is an informational rule, there are no potential issues related to this violation message.

### Consequences of Not Fixing

Since this is an informational rule, there is no implicit impact of this violation message

### How to Debug and Fix

The Soc\_04 rule assists in debugging the violations reported by the Soc\_01, Soc\_01\_info, Soc\_02 and Soc\_02\_info rules. The Soc\_04 rule is an informative rule and requires no debug.

No fix is required as this is an informational rule.

### Message 2

[INFO] Tag '<tag name>\_\_as\_used\_in\_scan\_chain\_tracing\_\_' is displayed for design '<du-name>'

### Arguments

- Name of the tag, <tag-name>
- Name of the design unit, <du-name>

### Potential Issues

A violation message is displayed for those [define\\_tag](#) tags that are used in [scan\\_chain](#) to define scan enable condition (-scanenable field).

### ***Consequences of Not Fixing***

Since this is an informational rule, there is no implicit impact of this violation message

### ***How to Debug and Fix***

The Soc\_04 rule assists in debugging the violations reported by the Soc\_01, Soc\_01\_info, Soc\_02 and Soc\_02\_info rules. The Soc\_04 rule is an informative rule and requires no debug.

No fix is required as this is an informational rule.

### **Example Code and/or Schematic**

Currently Unavailable

### **Default Severity Label**

Info

### **Rule Group**

SoC Rules

### **Reports and Related Files**

No related reports or files.

## Soc\_07

**Checks the structure between the user-specified nodes**

### When to Use

Use this rule when you want to use a specific structure.

### Description

The Soc\_07 rule checks whether the path all the -from nodes and at least one of the -from\_one\_of nodes to -to node have the same structure as specified in -type field of the *require\_structure* constraint. The Soc\_07 rule also allows functional buffers (1 input of OR/XOR gate is tied to 0, 1 input of AND is tied to 1). So, even if structure as specified in -type field acts as buffer, Soc\_07 considers them as valid objects on the path.

As an example of the rule's application, you can use this rule to run a check on memories. Memory-Controllers have PASS / FAIL status signals.

A GLOBAL-PASS is AND of all individual PASS signals as specified below:

```
require_structure -from "BIST::PASS" -to G_PASS -structure and
```

A GLOBAL-FAIL is OR of all individual FAIL signals as specified below:

```
require_structure -from "BIST::FAIL" -to G_FAIL -structure or
```

Based on the above status, we may have 3 types of failures:

- Extra driver is specified in the `from-node-list`, which is not present in the combinational fanin cone of `-to` node. However, the violation message for the extra driver is not reported for nodes given through `-from_one_of` field, if at least one of them is found in the fan in cone of the `-to` node.
- Missing driver where a node is in the combinational fanin cone of `to-node` but is missing from the `from-node-list`.
- Incorrect GATE is found in the combinational cloud between `from` and `to` nodes.

### Default Weight

10

## Language

Verilog, VHDL

## Parameter(s)

*dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

## Constraint(s)

*require\_structure* (mandatory): Use this constraint to define a structure check for all paths from source pins to destination pin.

## Operating Mode

None

## Messages and Suggested Fix

[WARNING] [constraint\_message\_tag: <value>]  
[Reason: <reason\_for\_violation>]Path from point '<from\_node>' to point '<to-node>' is not of type '<structure>'

### Arguments

- Constraint tag value, <value>
- One or combination of 'missing-driver', 'xtra-driver', incorrect-driver' may come as reason for violation, <reason\_for\_violation>
- Name of the source node, <from\_node>
- Name of the to destination node, <to\_node>
- Desired structure, <structure>

**NOTE:** The constraint tag value is prefixed to the violation message only if you specify the -constraint\_message\_tag argument for the *require\_structure* constraint.

### Potential Issues

The violation message appears if path between user-specified nodes is not user-specified.

### Consequences of Not Fixing

Not fixing this violation may result in error in evaluations in the circuit.



### How to Debug and Fix

Double-click the violation message. The Incremental Schematic window highlights the path between points specified in `-from` and `-to` field of the `require_structure` constraint.

To fix the violation, use user-specified structure only.

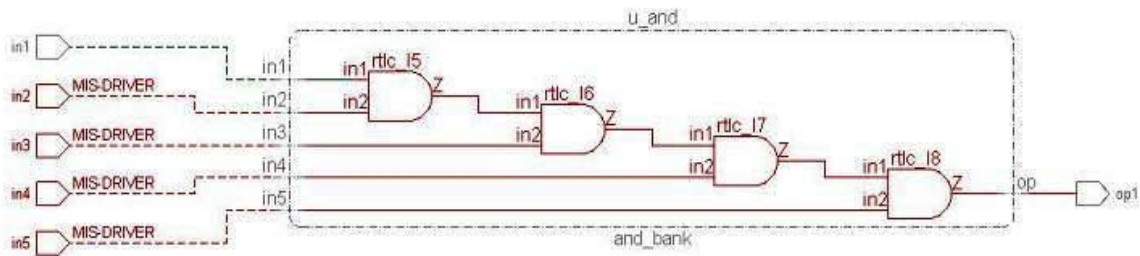
## Example Code and/or Schematic

### Example 1

Consider the following SGDC declaration:

```
require_structure -structure and -from in1 -to op1
```

Now, consider the following figure illustrating the Soc\_07 rule violation because of a missing driver:



For the above example, the Soc\_07 rule reports the following violation message:

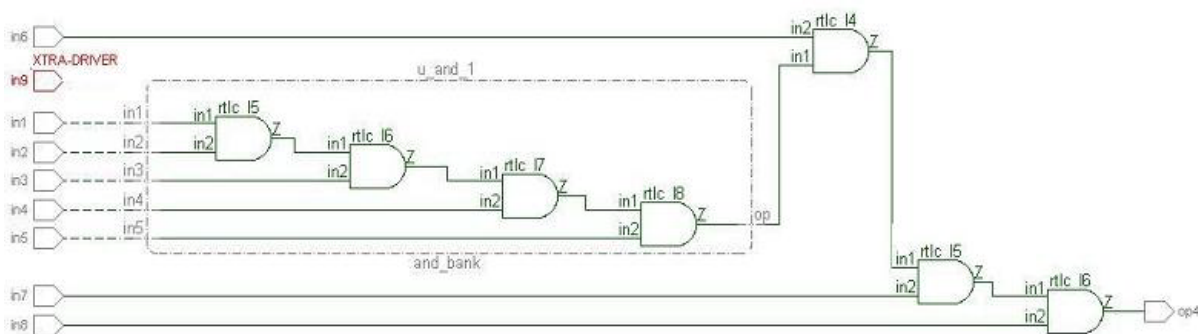
```
[Reason: missing-driver]Path from point 'in1' to point 'op1' is not of type 'and'
```

### Example 2

Consider the following SGDC declaration:

```
require_structure -structure and -from in1 in2 in3 in4 in5 in6 in7 in8 in9 -to op4
```

Consider the following figure illustrating the Soc\_07 rule violation because of an extra driver:



For the above example, the Soc\_07 rule reports the following violation message:

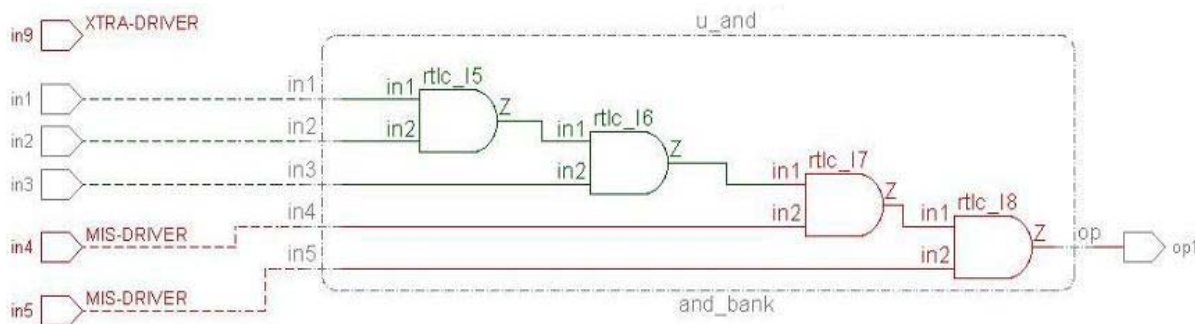
[Reason: xtra-driver]Path from point 'in9 in8 in7 in6 in5 in4 in3 in2 in1' to point 'op4' is not of type 'and'

### Example 3

Consider the following SGDC declaration:

```
require_structure -structure and -from in1 in2 in3 in9 -to op1
```

Consider the following figure illustrating the Soc\_07 rule violation because of extra and missing drivers:



For the above example, the Soc\_07 rule reports the following violation message:

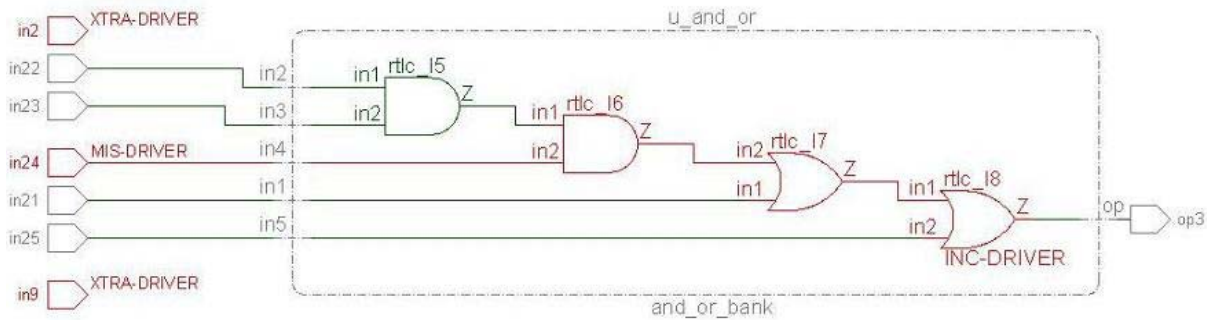
[Reason: xtra-driver, missing-driver]Path from point 'in9 in3 in2 in1' to point 'op1' is not of type 'and'

### Example 4

Consider the following SGDC declaration:

```
require_structure -structure and -from in21 in22 in23 in25
-to op3
```

Consider the following figure illustrating the Soc\_07 rule violation because of an extra, missing, and incorrect drivers:



For the above example, the Soc\_07 rule reports the following violation message:

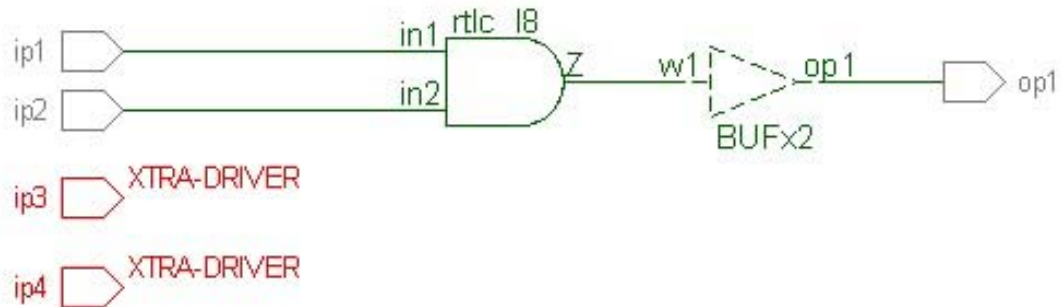
[Reason: xtra-dri ver, incorrect-dri ver, mi ssi ng-dri ver]Path from point 'in9 in2 in25 in23 in22 in21' to point 'op3' is not of type 'and'

### Example 5

Consider the following SGDC declaration:

```
require_structure -from ip1 ip2 -from_one_of ip3 ip4 -to op1
-structure and
```

Now, consider the following figure illustrating the Soc\_07 rule violation because of using the `-from_one_of` and `-structure` fields of the `require_structure` constraint simultaneously:



For the above example, the Soc\_07 rule reports the following violation message because none of the nodes specified using the `-from_one_of` field was found in the fanin of the `-to` node.

[Reason: xtra-driver]Path from point 'ip2 ip1, one of "ip4 ip3"' to point 'op1' is not of type 'and'

## Default Severity Label

Warning

## Rule Group

Soc

## Reports and Related Files

[dft\\_connectivity\\_check\\_summary.rpt](#): Reports the number of `require_structure` constraints passed and failed.

## Soc\_08

Checks the path between the user-specified nodes

### When to Use

Use this rule when you want to check the existence of a specific path.

### Description

The Soc\_08 rule checks if a path originating from the `-from` field to the `-to` field of the [require\\_strict\\_path](#) constraint exists in the design.

### Default Weight

10

### Language

Verilog, VHDL

### Parameter(s)

- [dftUseOffStateOfClockInClockPropagation](#): The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.
- [dft\\_allow\\_path\\_from\\_enable\\_to\\_cgc\\_clkout](#): The default value is off. Set the value of the parameter to on to allow a connectivity path from enable (data and test) to CGC clock-out pin.
- [dft\\_conn\\_check\\_allow\\_trace](#): The default value is off. Set the value of the parameter to yes to allow combinational traversal through the asynchronous pins of a flip-flop while performing the checks for require path ([require\\_path](#) constraint) or illegal path ([illegal\\_path](#) constraint).
- [dft\\_conn\\_check\\_handle\\_rtl\\_negedge](#): Default value is off. Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.
- [dft\\_require\\_path\\_fail\\_limit](#): The default value is 10. Set the value of the parameter to 10. Set the value of the parameter to any natural number to controls the number of violations reported by the Soc\_08 rule for the [require\\_strict\\_path](#) constraint failure when either the `-from_one_of` or `-to_one_of` arguments of the [require\\_strict\\_path](#) constraint is specified.

- *dft\_require\_path\_invalid\_limit*: The default value is 10. Set the value of the parameter to any natural number to control the number of invalid path violations reported by the *Soc\_08* rule for the *require\_strict\_path* constraint failure.
- *dft\_require\_path\_pass\_limit*: The default value is -1. Set the value of the parameter to any natural number to control the number of violations reported by the *Soc\_08* rule for the *require\_strict\_path* constraint failure when either the *-from\_one\_of* or *-to\_one\_of* arguments of the *require\_strict\_path* constraint is specified.
- *dft\_treat\_latches\_with\_X\_on\_enable\_as\_combinational\_for\_soc\_path\_checks*: The default value is off. Set the value of the parameter to on to define the treatment of latches. That is, whether to consider them as combinational or sequential, where enable pin does not get either 0 or 1, when running Soc path check rules.

## Constraint(s)

- *require\_strict\_path* (mandatory): Use this constraint to define a connectivity check for a path from a pin specified with the *-from* argument to a pin specified with the *-to* argument.
- *test\_mode* (optional): Use this constraint to specify the set of conditions, both pins and values, that when simulated, will force the circuit in test mode.
- *define\_tag* (optional): Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.

## Operating Mode

Scanshift, Capture, Define\_tag, Power Ground

## Messages and Suggested Fix

### Message 1

```
[INFO] [constraint_message_tag: <value>] Found valid path(s)
from '<from-node>' to '<to-nodes-count>' desired nodes <mode-
name>
```

### Arguments

To view the list of arguments, click [Arguments](#).

### ***Potential Issues***

Since this is an informational message, there are no potential issues related to this message.

### ***Consequences of Not Fixing***

Since this is an informational message, there is no implicit impact of this violation message.

### ***How to Debug and Fix***

No debug or fix is required as this is an informational message.

### **Message 2**

[WARNING] [constraint\_message\_tag: <value>] Found path from '<from-node>' to '<to-nodes-count>' undesired node <mode-name>

### ***Arguments***

The node at which the traversal was stopped, <to-node>

To view the other arguments, click [Arguments](#).

### ***Potential Issues***

The violation is reported because a valid path to a design node, which was not specified in the -to field of the require\_strict\_path constraint, is found by SpyGlass.

### ***Consequences of Not Fixing***

Not fixing this violation may result in unexpected results.

### ***How to Debug and Fix***

Double-click the violation message to open the spreadsheet, which contains the list of all the violation messages. Click on a violation message in the spreadsheet to view the incremental schematic of the violation message. The incremental schematic displays the path specified in the violation message.

To fix the violation, verify the highlighted path. If the path is correct, add the <to-node> node in the -to field of the require\_strict\_path constraint in the SGDC file. However, if the path is not correct, specify any missing testmode constraint or check the design.

### **Message 3**

[WARNING] [constraint\_message\_tag: <value>] No Valid path(s)

from '<from-node>' to '<to-nodes-count>' desired nodes <mode-name>

### **Arguments**

- Constraint tag value, <value>
- Source node from which the connectivity check is performed, <from-node>
- Number of <to-nodes> at which the traversal was stopped, <to-nodes-count>

Please note that this list may or may not be present in the -to field of the `require_strict_path` constraint.

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the -constraint\_message\_tag argument for the [require\\_strict\\_path](#) constraint.*

- The mode name, <mode-name>

The available mode names are specified in the following table:

Specified Field of the <code>require_strict_path</code> Constraint	Corresponding <mode-name>
-use_shift	Shift
-use_capture	Capture
-use_captureATspeed	Capture (atspeed)
-tag	Define_tag
If none of the above fields is specified	Power Ground

### **Potential Issues**

The violation message appears if no valid path exists between the user-specified nodes.

### **Consequences of Not Fixing**

Not fixing this violation may impact the functionality of the design.

### **How to Debug and Fix**

To fix this violation, perform any of the following steps:

- Check the design for an error
- Check the SGDC file for a missing constraint



**Message 4**

**[WARNING]** [constraint\_message\_tag: <value>] '<no\_of\_paths>' (<limit\_breached>) path(s) found from '<from\_node\_name>' to desired nodes in <simulation condition>

**Arguments**

- Constraint tag value, <value>
- Limit to the number of paths, <limit\_breached>
- Node name specified in the '-from' field of 'require\_strict\_path' constraint, <from\_node\_name>
- Simulation condition specified in the 'require\_strict\_path' constraint, <simulation\_condition>

**Potential Issues**

Specified number of paths are not present between user-specified nodes.

**Consequences of Not Fixing**

Not fixing the violation may result in unsearchable expected path. This may impact the functionality of the design.

**How to Debug and Fix**

Double-click the violation message to open the spreadsheet, which contains the list of all the violation messages. Click on a violation message in the spreadsheet to view the incremental schematic of the violation message. The incremental schematic displays the path specified in the violation message.

If endpoints are displayed start probing from either fan-out cone of from point or fan-in cone of to point to see why required path does not exist.

If the path from start point up to the instance which is blocking is displayed then analyze the text annotation date displayed to see why the path is blocked at the instance highlighted.

To fix this violation, perform any of the following steps:

- Check the design for an error
- Check the SGDC file for a missing constraint

**Message 5**

**[WARNING]** [constraint\_message\_tag: <value>] min\_to\_paths('<min\_path\_value>') should have a value less than

`max_to_paths(' <max_path_value>')`, ignoring these values

### **Arguments**

- Constraint tag value, <value>
- Minimum number of expected successful paths, <min\_path\_value>
- Maximum number of expected successful paths, <max\_path\_value>

### **Potential issues**

The value specified is ignored.

### **How to debug and fix**

Double-click the violation message to open the spreadsheet, which contains the list of all the violation messages. Click on a violation message in the spreadsheet to view the incremental schematic of the violation message. The incremental schematic displays the path specified in the violation message.

If endpoints are displayed start probing from either fan-out cone of from point or fan-in cone of to point to see why required path does not exist.

If the path from start point up to the instance which is blocking is displayed then analyze the text annotation date displayed to see why the path is blocked at the instance highlighted.

To fix the violation, review the constraint and modify to specify valid values.

## **Example Code and/or Schematic**

Consider the following SGDC file snippet:

```
current_design dt
define_tag -tag mbel -name bbox.en -value 1
testmode -name bbox.en -value 0
require_strict_path -from "Sflop.*reg.Q" -to "flop2.*_reg.D"
-use_shift
```

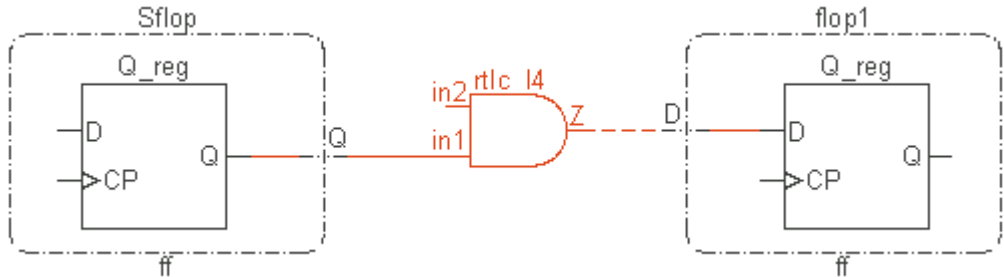
Now, consider the following violation reported by the Soc\_08 rule for the above specification of the `require_strict_path` constraint:

**[WARNING]** Found valid path from 'dt.Sflop.Q\_reg.Q' to undesired node 'dt.flop1.Q\_reg.D' in shift mode

The above violation is reported because there exists a valid path from the

dt.Sflop.Q\_reg.Q node to the dt.flop1.Q\_reg.D node, however, the dt.flop1.Q\_reg.D node is not specified in the `-to` field of the above `require_strict_path` constraint specification.

The following schematic corresponds to the above violation message:



## Default Severity Label

Info/Warning

## Rule Group

SoC

## Reports and Related Files

[dft\\_connectivity\\_check\\_summary.rpt](#): Reports the number of `require_strict_path` constraints passed and failed.

## Soc\_09

### Path between user-specified nodes should not exist

#### When to Use

Use this rule to ensure that the path between two nodes does not exist.

#### Description

The Soc\_09 rule reports a violation, if a path exists between from and to nodes of the *illegal\_path* constraint.

The type of path to be searched is determined by the `-path_type` argument of the *illegal\_path* constraint. This argument can take one of the following values: *buffered*, *sensitized*, or *sensitizable*. By default, sensitizable path is searched.

#### Default Weight

10

#### Language

Verilog, VHDL

#### Parameter(s)

- *dft\_treat\_latches\_with\_X\_on\_enable\_as\_combinational\_for\_soc\_path\_checks*: The default value is off. Set the value of the parameter to on to define the treatment of latches. That is, whether to consider them as combinational or sequential, where enable pin does not get either 0 or 1, when running Soc path check rules.
- *dft\_allow\_path\_from\_enable\_to\_cgc\_clkout*: The default value is off. Set the value of the parameter to on to allow a connectivity path from enable (data and test) to CGC clock-out pin.
- *dft\_conn\_check\_allow\_trace*: The default value is off. Set the value of the parameter to on to allow combinational traversal through the asynchronous pins of a flip-flop while performing the checks for require path (*require\_path* constraint) or illegal path (*illegal\_path* constraint).
- *dft\_conn\_check\_handle\_rtl\_negedge*: Default value is off. Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.

## Constraint(s)

- *illegal\_path* (mandatory): Use this constraint to define nodes between which path should not exist.
- *test\_mode* (optional): Use this constraint to specify the set of conditions, both pins and values, that when simulated, will force the circuit in test mode.
- *define\_tag* (optional): Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.

## Operating Mode

Scanshift, Capture, Define\_tag, Power Ground

## Messages and Suggested Fix

### Message 1

[ERROR] [constraint\_message\_tag: <value>] '<path\_type>' path found from '<from\_node>' to undesired node '<to\_node>' in '<mode\_name>'

### Arguments

- Type of path found, <path\_type>
- From node name, <from\_node>
- To node name, <to\_node>
- Simulation condition, <mode\_name>

### Potential Issues

The violation message is reported because a valid path exists from <from\_node> to <to\_node>, which are specified using the *illegal\_path* constraint.

### Consequences of Not Fixing

Not fixing this violation may impact the functionality of design.

### How to Debug and Fix

View the incremental schematic of the violation message. It displays the path specified in the violation message.

To fix the violation, verify the highlighted path. If the path is correct,

modify the corresponding *illegal\_path* constraint. However, if the path is not correct, specify any missing *test\_mode/define\_tag* constraint or check the design.

## Message 2

**[ERROR]** [constraint\_message\_tag: <value>]' <path\_type>' path found to '<to\_node>' from undesired node '<from\_node>' in '<mode\_name>'

### Arguments

- Type of path found, <path\_type>
- From node name, <from\_node>
- To node name, <to\_node>
- Simulation condition, <mode\_name>

### Potential Issues

The violation message is reported because a valid path exists to <to\_node> from <from\_node>, which are specified using the *illegal\_path* constraint.

### Consequences of Not Fixing

Not fixing this violation may impact the functionality of design.

### How to Debug and Fix

View the incremental schematic of the violation message. It displays the path specified in the violation message.

To fix the violation, verify the highlighted path. If the path is correct, modify the corresponding *illegal\_path* constraint. However, if the path is not correct, specify any missing *test\_mode/define\_tag* constraint or check the design.

## Message 3

**[ERROR]** [constraint\_message\_tag: <value>]' <From | To>' node <node name> is connected to leaf cell(s)

### Arguments

- Constraint tag value, <value>
- From / To node name, <node name>

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the*

-*constraint\_message\_tag* argument for the *illegal\_path* constraint.

### **Potential Issues**

The violation message is reported because a from node was driving a leaf cell or a to node was driven by a leaf cell. The violation message is reported only for to or from nodes which were specified without any from or to nodes, respectively, in the *illegal\_path* constraint.

### **Consequences of Not Fixing**

Not fixing this violation may impact the functionality of design.

### **How to Debug and Fix**

View the incremental schematic of the violation message. It displays the path specified in the violation message.

To fix the violation, verify the highlighted path. If the path is correct, modify the corresponding *illegal\_path* constraint. However, if the path is not correct, specify any missing *test\_mode/define\_tag* constraint or check the design.

### **Message 4**

```
[INFO] [constraint_message_tag: <value>] ' <passed_checks>' out
of ' <total_checks>' check(s) passed
```

### **Potential Issues**

Not Applicable.

### **Consequences of Not Fixing**

Not Applicable.

### **How to Debug and Fix**

Not Applicable.

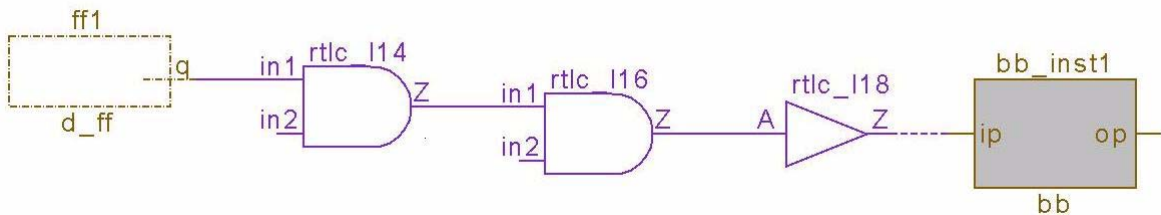
## **Example Code and/or Schematic**

### **Example 1**

Consider the following sample SGDC file snippet:

```
current_design test.behaviour
illegal_path -from test.ff1.q -to test.bb_inst1.ip
-use_shift
```

Now, consider the following figure:



The Soc\_09 rule reports the following violation message for the above specification of the *illegal\_path* constraint:

```
[ERROR] 'Sensitive' path found from 'test.ff1.q' to
undesired node 'test.bb_inst1.ip' in 'Shift mode'
```

The rule reports the above violation message because there exists a sensitizable path from the test.ff1.q node to the test.bb\_inst1.ip node.

### Example 2

Consider the following violation message reported by the Soc\_09 rule:

```
[constraint_message_tag: ip_A_to_B] No 'Sensitive' path
found from '6' (out of '12') From-Nodes to some (out of '10')
To-Nodes in 'Power-Ground mode', SPREADSHEET_PATH:
'spyglass_reports/dft/Soc_09/000/Soc_09_PASS_001.csv'
```

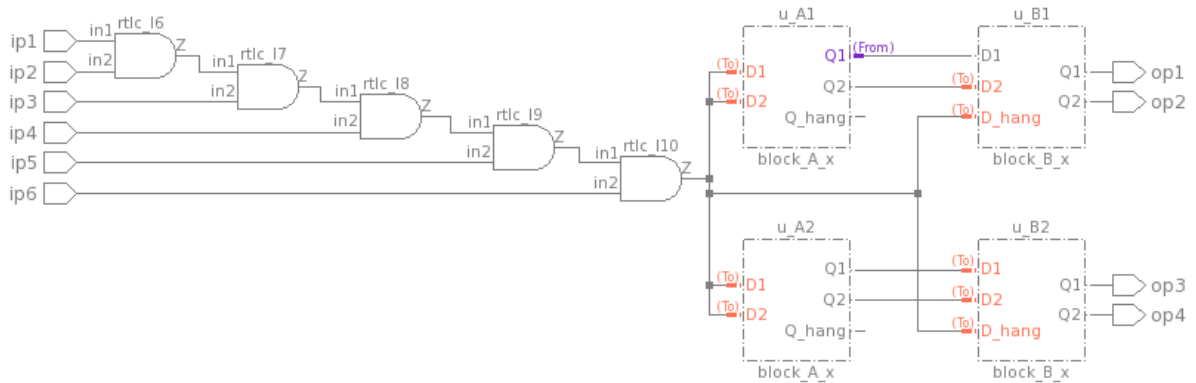
Double-click on the violation message to view the related spreadsheet report as shown in the following figure:

B	C
"From-Point"	"Count of unreachable End-Points"
top.u_A1.Q1	9
top.u_A1.Q2	9
top.u_A1.Q_hang	10
top.u_A2.Q1	9
top.u_A2.Q2	9
top.u_A2.Q_hang	10

Each row of the Spreadsheet report contains the name of the From-Node and count of the To-Nodes, for which path is not found. Click on a row to view the corresponding Schematic, displaying the From-Node and all the unreachable To-Nodes, as shown in the following figure:



## Overview



Also, the Incremental Schematic of the main violation shows all the From-Nodes for which, at least one path to illegal end points was not found.

## Default Severity Label

Error

## Rule Group

SoC

## Reports and Related Files

[dft\\_connectivity\\_check\\_summary.rpt](#): Reports the number of *illegal\_path* constraints passed and failed.

## Soc\_01\_Info

**Displays information for node whose expected node value is achieved.**

### When to Use

Use this rule to display the nodes that are specified with the `require_value` constraint and have the same simulation value when the specified tag condition is simulated.

### Description

The Soc\_01\_Info rule generates the SpyGlass Explorer highlight data for those nodes specified with `require_value` constraints that achieve the specified simulation value when the specified tag condition is simulated.

Consider the following `require_value` constraint:

```
require_value
  -tag <tagName> -name <nodeNames> -value <value>
```

**NOTE:** To view the nodes that do not achieve the expected value, use the [Soc\\_01](#) rule.

### Prerequisites

Specify the `require_value` constraint.

### Default Weight

10

### Language

Verilog, VHDL

### Method

For each `define_tag`, simulate power, ground and all conditions defined for this `define_tag`.

For each `require_value` with the current `define_tag`, check that the defined pin has the defined value. If expected value is achieved, report a message.

### Parameter(s)

- `dft_conn_check_handle_rtl_negedge`: Default value is off. Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.

- *dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

## Constraint(s)

- *require\_value* (mandatory): Use this constraint to define a connectivity check for a path from a pin specified with the -from argument to a pin specified with the -to argument.
- *define\_tag* (optional): Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.

## Operating Mode

Define\_tag

## Messages and Suggested Fix

### Message 1

The Soc\_01\_Info rule generates the SpyGlass Explorer highlight data for those nodes specified with the -name argument that achieve the simulation value `<value>` when the conditions of `<tagName>` are simulated.

For each such node, the following message is generated:

```
[INFO] [constraint_message_tag: <value>] Node '<node-name>' has
required value(<value>) under tag '<tagName>'
```

Where `<node-name>` is a node specified with the -name argument.

### Arguments

- Constraint tag value, `<value>`
- Name of the node. `<node-name>`
- Simulation value. `<value>`
- Tag name. `<tagName>`

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the -constraint\_message\_tag argument for the [require\\_value](#) constraint.*

### Potential Issues

This is an informational rule. Therefore, it does not have any related potential issues.

### ***Consequences of Not Fixing***

This is an informational rule. Therefore, it does not have any implicit impact.

### ***How to Debug and Fix***

View the Incremental Schematic of the violation message. The Incremental Schematic displays the node where the required simulation matches the actual simulation.

You can also view the violation for the Soc\_04 rule along with the violation of the Soc\_01\_Info rule in the Incremental Schematic window. To do this, double-click the violation for the Soc\_01\_Info rule and open the Incremental Schematic window.

The violation message for the Soc\_04 rule overlaps the violation message for the Soc\_01\_Info rule in the Incremental Schematic window. This is useful in debugging the violation for the Soc\_01\_Info rule.

To fix the violation, see Example Code and/or Schematic section.

## **Message 2**

**[WARNING]** [constraint\_message\_tag: <value>] illegal\_value command is using option -matchNBits (<value1>) without setting parameter dftShowWaveForm to 'on' under <tag-name>. Setting -matchNBits to value '1' for the rule checking purpose

### ***Arguments***

- Constraint tag value, <value>
- Illegal value <value1>
- Tag name <tag-name>

### ***Potential Issues***

A violation is reported when the following conditions hold true:

- Value of the *dftShowWaveForm* parameter is not set on
- The *require\_value* constraint uses the -matchNbits <value> argument under <tag-name> condition
- The value of the -matchNBits argument is greater than 1

The *Soc\_01\_Info* rule does not report this violation message if the tag is

## Overview

defined using the `define_tag` command.

### ***Consequences of Not Fixing***

Not fixing the violation may result in unexpected code behavior.

### ***How to Debug and Fix***

To fix the violation, set the value of the `dftShowWaveForm` parameter to on

## **Example Code and/or Schematic**

Consider the following example:

The screenshot displays the Atrenta Console interface for a project named 'Project-1.prj'. The main window shows an 'Incremental Schematic : tap' with a circuit diagram. The schematic includes a dashed box labeled 'INSTR\_DEC' containing two multiplexers (rtl\_c\_l18 and rtl\_c\_l16) and a register (mode0\_reg). The register's output is connected to the select input of rtl\_c\_l16. The output of rtl\_c\_l16 is connected to the select input of rtl\_c\_l18. The output of rtl\_c\_l18 is connected to the 'mode1' output. The output of mode0\_reg is connected to the 'mode0' output. The 'mode0' output is also connected to the 'mode0' output of rtl\_c\_l16. The 'mode1' output is connected to the 'mode1' output of rtl\_c\_l18. The 'mode0' output is connected to the 'mode0' output of rtl\_c\_l16. The 'mode1' output is connected to the 'mode1' output of rtl\_c\_l18.

The bottom panel shows a 'View: Msg' window with the following messages:

```

ERROR: [ 1 ]
 Soc_01[1]:Expected node value must be achieved.
  Node 'mode0' has value 0 (Required: 1) under tag '-useTestmode',
  tap.sgdc, 25
INFO: [ 5 ]
 Soc_01_Info[3]:Expected node value is achieved.
  Node 'mode1' has required value(1) under tag '-useTestmode', tap.sgdc, 26
  Node 'mode2' has required value(0) under tag '-useTestmode', tap.sgdc, 27
  
```

## Default Severity Label

Info

## Rule Group

SoC

## Reports and Related Files

[dft\\_connectivity\\_check\\_summary.rpt](#): Reports the number of [require\\_value](#) constraints passed and failed.

## Soc\_02\_Info

**Displays information for the connected user-specified nodes.**

### When to Use

Use this rule to generate information for the connected user-specified nodes.

### Rule Description

The Soc\_02\_Info rule generates the SpyGlass Explorer highlight data for valid paths specified with *require\_path* constraints.

To view the invalid paths, use the *Soc\_02* rule.

While processing the *require\_path* constraints, Soc\_02\_Info rule checking depends on *-path\_type* and simulation condition as discussed below:

Specified field of <i>require_path</i> constraint	Simulation condition
<i>-use_shift</i>	Shift
<i>-use_capture</i>	Capture
<i>-use_captureATspeed</i>	Capture (atspeed)
<i>-tag &lt;tag_name&gt;</i>	tag_name
If none of the above specified	Power ground

- If you specify the value of the *-path\_type* argument as sensitized, the Soc\_02\_Info rule performs the strict functional checking. This ensures that the path is properly sensitized by the specified simulation condition.
- If you specify the value of the *-path\_type* argument as sensitizable, the Soc\_02\_Info rule performs the functional checking. This ensures that the path is properly sensitizable and is not blocked by the specified simulation condition.
- If you specify the value of the *-path\_type* argument as buffered, the Soc\_02\_Info rule looks for strict topological checking and checks for buffers and inverters only.
- If you do not specify the value of the *-path\_type* argument, the default value of the argument, that is, sensitizable is used.

**NOTE:** *If you do not specify the -undirected qualifier in the require\_path constraint, the Soc\_02\_Info rule traces the path from -from node to -to node only.*

## Prerequisites

Specify the *require\_path* constraint.

## Default Weight

10

## Language

Verilog, VHDL

## Method

If specified, simulate require path for <path\_type> verification under specified simulation condition.

For each from-to pair of *require\_path* specification:

Walk the unblocked fan-out cone under the simulation condition from the start point while maintaining the phase inversion. If the walk terminates on the specified endpoint for the required <pathtype>, report a violation.

## Parameter(s)

- *dftAllowNonXValueAtStartOfSensitizedPathInSoc\_02*: This parameter is deprecated and is ignored for rule-checking.
- *dft\_allow\_path\_from\_enable\_to\_cgc\_clkout*: The default value is off. Set the value of the parameter to on to allow a connectivity path from enable (data and test) to CGC clock-out pin.
- *dft\_conn\_check\_allow\_non\_x\_value\_on\_sensitizable\_path*: The default value is on. Set the value of the parameter to off to ignore non-X (0 or 1) value on the path while performing checks on the sensitizable paths.
- *dft\_conn\_check\_handle\_rtl\_negedge*: Default value is off. Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.
- *dft\_require\_path\_pass\_limit*: The default value is -1. Set the value of the parameter to any natural number to control the number of violations reported by the *Soc\_02\_Info* rule for the *require\_path* constraint success when either the *-from\_one\_of* or *-to\_one\_of* arguments of the *require\_path* constraint is specified.
- *dft\_require\_path\_stop\_check\_on\_pass\_count*: The default value is -1. Set the value of the parameter to any natural number to control the number of violations reported by the *Soc\_02\_Info* rule for the *require\_path*



constraint success when either the `-from_one_of` or `-to_one_of` arguments of the `require_path` constraint is specified.

- *dft\_treat\_latches\_with\_X\_on\_enable\_as\_combinational\_for\_soc\_path\_checks*: The default value is off. Set the value of the parameter to on to define the treatment of latches. That is, whether to consider them as combinational or sequential, where enable pin does not get either 0 or 1, when running Soc path check rules.
- *dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.
- *dft\_conn\_check\_allow\_trace*: The default value is off. Set the value of the parameter to on to allow combinational traversal through the asynchronous pins of a flip-flop while performing the checks for `require_path` (*require\_path* constraint) or illegal path (*illegal\_path* constraint).

## Constraint(s)

- *require\_path* (mandatory): Use this constraint to define a connectivity check for a path from a pin specified with the `-from` argument to a pin specified with the `-to` argument.

If you do not specify the `-undirected` qualifier in the `require_path` constraint, the `Soc_02_Info` rule traces the path from `-from` node to `-to` node only.

- *test\_mode* (optional): Use this constraint to specify the set of conditions, both pins and values, that when simulated, will force the circuit in test mode.
- *define\_tag* (optional): Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.

## Operating Mode

Scanshift, Capture, Power Ground, Define\_tag

## Messages and Suggested Fix

### Message 1

[ERROR] [constraint\_message\_tag: <value>] ' <path\_type>' path(s)

found from '<from\_node\_name>' to '<to\_node\_count>' nodes under <simulation condition> within '<sequential\_depth>' sequential depth

### **Arguments**

To view the list of message arguments, click [Arguments](#).

### **Potential Issues**

Since this is an informational rule, there are no potential issues related to this violation message.

### **Consequences of Not Fixing**

Since this is an informational rule, there is no implicit impact of this violation message.

### **How to Debug and Fix**

For more information on debugging and fixing the violation, click [How to Debug and Fix](#).

### **Message 2**

[ERROR] [constraint\_message\_tag: <value>] '<path\_type>' path(s) found to '<to\_node\_name>' from '<from\_node\_count>' nodes under <simulation condition> within '<sequential\_depth>' sequential depth

### **Arguments**

To view the list of message arguments, click [Arguments](#).

### **Potential Issues**

Since this is an informational rule, there are no potential issues related to this violation message.

**NOTE:** *this violation message is reported instead of Message-1, in case of '-from\_one\_of' field specified in the require\_path constraint.*

### **Consequences of Not Fixing**

Since this is an informational rule, there is no implicit impact of this violation message.

### **How to Debug and Fix**

For more information on debugging and fixing the violation, click [How to Debug and Fix](#).

### **Message 3**

**[ERROR]** [constraint\_message\_tag: <value>] ' <path\_type>' path(s) found from ' <from\_node\_name>' to ' <to\_node\_count>' nodes within ' <sequential\_depth >' sequential depth

### **Arguments**

To view the list of message arguments, click [Arguments](#).

### **Potential Issues**

Since this is an informational rule, there are no potential issues related to this violation message.

**NOTE:** *This violation is reported instead of Message-1, when '-path\_type' is 'buffered' in 'require\_path' constraint.*

### **Consequences of Not Fixing**

Since this is an informational rule, there is no implicit impact of this violation message.

### **How to Debug and Fix**

For more information on debugging and fixing the violation, click [How to Debug and Fix](#).

### **Message 4**

**[ERROR]** [constraint\_message\_tag: <value>] ' <path\_type>' path(s) not found to ' <to\_node\_name>' from ' <from\_node\_count>' nodes within ' <sequential\_depth >' sequential depth

### **Arguments**

- Constraint tag value, <value>
- Expected path type, as specified by 'require\_path' constraint, <path\_type>
- Node name specified in the '-from' field of 'require\_path' constraint, <from\_node\_name>
- Node name specified in the '-to' field of 'require\_path' constraint, <to\_node\_name>
- No of '-to' nodes for which connectivity check failed from the specified '-from' node, <to\_node\_count>
- No of '-from' nodes from which connectivity check failed to the specified '-to' node, <from\_node\_count>

- Value specified by '-sequential\_depth' field of 'require\_path' constraint, <sequential\_depth>
- Simulation condition specified in the 'require\_path' constraint, <simulation\_condition>

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the -constraint\_message\_tag argument for the [require\\_path](#) constraint.*

### **Potential Issues**

Since this is an informational rule, there are no potential issues related to this violation message.

**NOTE:** *This violation message is reported instead of Message-2, when '-path\_type' is 'buffered' in 'require\_path' constraint.*

### **Consequences of Not Fixing**

Since this is an informational rule, there is no implicit impact of this violation message.

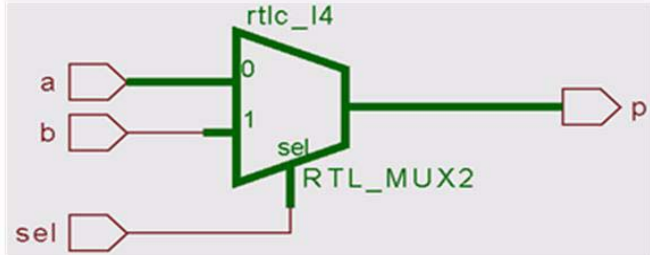
### **How to Debug and Fix**

Double-click the violation message to open the spreadsheet, which contains the list of all the violation messages. Click on a violation message in the spreadsheet to view the incremental schematic of the violation message. The incremental schematic displays the path specified in the violation message.

## **Example Code and/or Schematic**

Consider the following example:

```
// File: test.sgdc
current_design test
require_path -from a -to p
```



Soc\_02\_Info[1]:Connection between user specified nodes exists.  
 Path from point 'a' to point 'p' is 'sensitizable' under power ground simulation condition, test.sgdc, 2

### Schematic Highlight

Path between the points specified with the `-from` and `-to` fields of the `require_path` constraint.

### Default Severity Label

Info

### Rule Group

SoC

### Reports and Related Files

[dft\\_connectivity\\_check\\_summary.rpt](#): Reports the number of `require_path` constraints passed and failed.

## Soc\_07\_Info

**Reports the existence of structure between user-specified nodes**

### When to Use

Use this rule to confirm if the user-specified structure is used.

### Description

The Soc\_07\_Info rule displays all the paths from -from nodes and at least one of the -from\_one\_of nodes to -to node that have the same structure as specified in -type field of the *require\_structure* constraint. The Soc\_07\_Info rule also displays paths which act as functional buffers (1 input of OR/XOR gate is tied to 0, 1 input of AND is tied to 1)

As an example of the rule's application, you can use this rule to run a check on memories. Memory-Controllers have PASS / FAIL status signals.

A GLOBAL-PASS is AND of all individual PASS signals as specified below:

```
require_structure -from "BIST::PASS" -to G_PASS -structure and
```

A GLOBAL-FAIL is OR of all individual FAIL signals as specified below:

```
require_structure -from "BIST::FAIL" -to G_FAIL -structure or
```

Based on the above status, we may have 3 types of failures:

- Extra driver is specified in the `from-node-list`, which is not present in the combinational fanin cone of -to node. However, the violation message for the extra driver is not reported for nodes given through -from\_one\_of field, if at least one of them is found in the fan in cone of the -to node.
- Missing driver where a node is in the combinational fanin cone of to-node but is missing from the `from-node-list`.
- Incorrect GATE is found in the combinational cloud between from and to nodes.

### Default Weight

10

### Language

Verilog, VHDL

## Parameter(s)

*dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

## Constraint(s)

*require\_structure* (Mandatory): Use this constraint to define a structure check for all paths from source pins to destination pin.

## Operating Mode

None

## Messages and Suggested Fix

[INFO] [constraint\_message\_tag: <value>] Path from point '<from\_node>' to point '<to\_node>' is of type '<structure>'

### Arguments

- Constraint tag value, <value>
- Name of the source node, <from\_node>
- Name of the to destination node, <to\_node>
- Desired structure, <structure>

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the -constraint\_message\_tag argument for the [require\\_structure](#) constraint.*

### Potential Issues

The violation message appears when the structure between two points matches with the user-specified structure.

### Consequences of Not Fixing

This is an informational rule. Therefore, there are no direct consequences of not fixing this violation message.

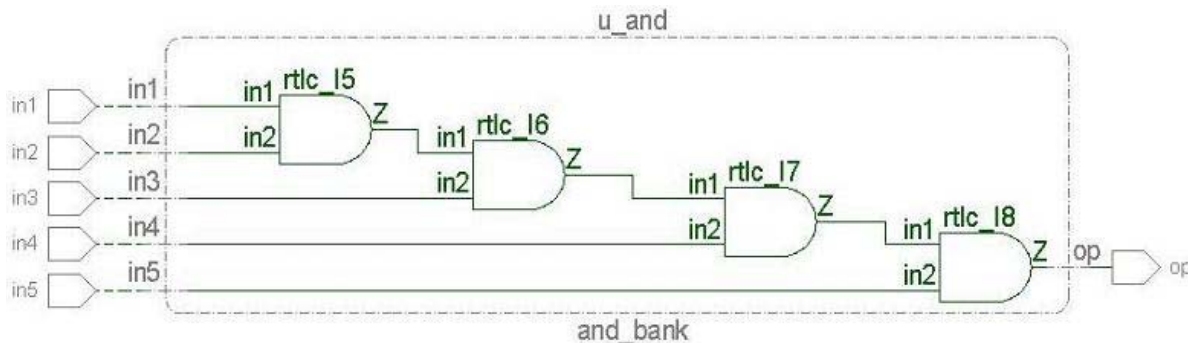
### How to Debug and Fix

This is an informational rule. Therefore, no debug or fix information is required for this violation message.

## Example Code and/or Schematic

### Example 1

Consider the following figure:



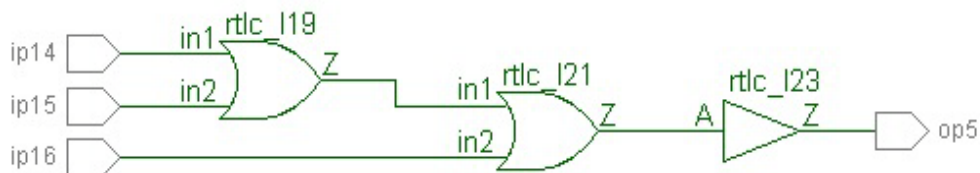
For the above example, the Soc\_07\_Info rule reports an information message stating that structure specified is a valid structure.

### Example 2

Consider the following SGDC declaration:

```
require_structure -from ip14 ip15 -from_one_of ip16 ip17 -to
op5 -structure or
```

Now, consider the following schematic:



The Soc\_07\_Info rule reports the following violation message for the above schematic:

Path from point 'ip15 ip14, one of "ip17 ip16"' to point 'op5' is of type 'or'



## Default Severity Label

Info

## Rule Group

SoC

## Reports and Related Files

[dft\\_connectivity\\_check\\_summary.rpt](#): Reports the number of [require\\_structure](#) constraints passed and failed.

## Soc\_10

### Reports nets with illegal node values

#### When to Use

Use this rule to identify the nodes that have a user-specified value under the specified simulation condition.

#### Description

The *Soc\_10* rule generates the SpyGlass Explorer highlight data for those nodes specified with *illegal\_value* constraints that contain the illegal simulation value when the specified tag condition is simulated.

Expected values at arbitrary nodes and the applied values that should cause or force the expected values are checked. This is useful for ensuring that unit-level test requirements are satisfied at the SoC level.

Consider the following *illegal\_value* constraint:

```
illegal_value
  -tag <tagName> -name <nodeNames> -value <value>
```

The *Soc\_10* rule generates the SpyGlass Explorer highlight data for those nodes specified with the `-name` argument get value `<value>` when the conditions specified by `<tagName>` are simulated.

#### Prerequisites

Specify the *illegal\_value* constraint.

#### Default Weight

10

#### Language

Verilog, VHDL

#### Parameter(s)

- *dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

- *dft\_conn\_check\_handle\_rtl\_negedge*: Default value is off. Set the value of the parameter to yes to consider the input to the inverter, in front of the CP/CLR/PRE pin, as the start/end point of the connectivity check.

## Constraint(s)

- *define\_tag*: Use this constraint to define a named condition for application of certain stimulus at the top port or an internal node.
- *illegal\_value*: Use this constraint to check for the presence of an illegal value on a certain node when the circuit has been simulated using the condition specified by the -tag argument.

## Operating Mode

Define\_tag

## Messages and Suggested Fix

The following violation messages are reported by the Soc\_10 rule:

### Message 1

[ERROR] [constraint\_message\_tag: <value>] Node <name> has illegal value <value1>. under tag <tag-name>

### Arguments

- Constraint tag value, <value>
- Name of the node <name>
- Illegal value <value1>
- Tag name <tag-name>

**NOTE:** *The constraint tag value is prefixed to the violation message only if you specify the -constraint\_message\_tag argument for the *illegal\_value* constraint.*

### Potential Issues

A violation is reported due to incomplete or incorrect simulation condition or incorrect design connectivity.

### Consequences of Not Fixing

Not fixing the violation may result in unexpected code behavior.

### How to Debug and Fix

View the Incremental Schematic for the violation message. The

Incremental Schematic highlights the node with the illegal value at the simulation.

You can also view the violation for the Soc\_04 rule along with the violation of the Soc\_10 rule in the Incremental Schematic window. To do this, double-click the violation for the Soc\_10 rule and open the Incremental Schematic window.

The violation message for the Soc\_04 rule overlaps the violation message for the Soc\_10 rule in the Incremental Schematic window. This is useful in debugging the violation for the Soc\_10 rule.

To fix the violation, see [Example Code and/or Schematic](#) section.

## Message 2

**[WARNING]** [constraint\_message\_tag: <value>] illegal\_value command is using option -matchNBits (<value1>) without setting parameter dftShowWaveForm to 'on' under <tag-name>. Setting -matchNBits to value '1' for the rule checking purpose

### Arguments

- Constraint tag value, <value>
- Illegal value <value1>
- Tag name <tag-name>

### Potential Issues

This violation is reported when the following conditions hold true:

- Value of the [dftShowWaveForm](#) parameter is not set to on
- The [illegal\\_value](#) constraint uses the -matchNBits <value> argument under <tag-name> condition. If you have not specified the -matchNBits argument:
  - under a tag defined using define\_tag constraint, all bits in the sequence are checked
  - under use\_shift, use\_capture, or use\_captureATspeed, the last bit value is checked. See [Example 3](#) for more information.
- The value of the -matchNBits argument is greater than 1

The Soc\_10 rule does not report this violation message if the tag is defined using the [define\\_tag](#) command.

### Consequences of Not Fixing

Not fixing the violation may result in unexpected code behavior.

### ***How to Debug and Fix***

To fix the violation, set the value of the *dftShowWaveForm* parameter to on.

### **Message 3**

```
[INFO] <constraint_message_tag>Node ' <node_name>' has valid value ' <current_value> (illegal: <disallowed_value>)' under <mode>
```

### ***Arguments***

- Name of the constraint\_message\_tag, if present, <constraint\_message\_tag>
- Name of the design node which does not have the illegal value (check PASSED), <node\_name>
- Current value, <current\_value>
- Disallowed value, <disallowed\_value>
- Name of the 'mode' (simulation condition), <mode>

### ***Potential Issues***

This is an informational message. Therefore, there are no potential issues related to this message.

### ***Consequences of Not Fixing***

This is an informational message. Therefore, there are no consequences of not fixing this message.

### ***How to Debug and Fix***

This is an informational message. Therefore, no debug or fix is required.

## **Example Code and/or Schematic**

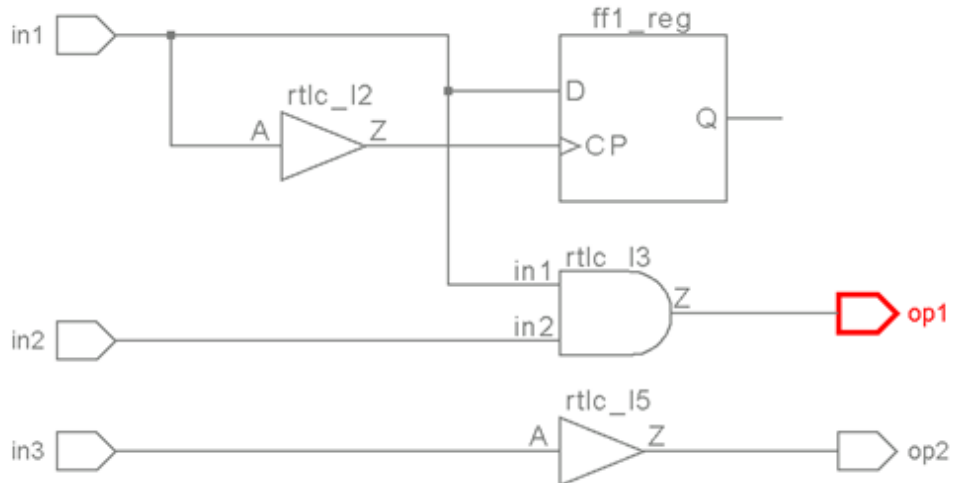
### **Example 1**

Consider the following sample SGDC file describing the *illegal\_value* constraint description:

```
current_design top
  illegal_value -name top.op1 -value 1 -use_shift
  test_mode -name in1 -value 1
  test_mode -name in2 -value 1
```

*Figure 1* describes the schematic depicting the violation displayed for the

*illegal\_value* constraint:



**FIGURE 1.** Violation for the *illegal\_value* constraint

In the above design, in test mode the value at `top.op1` is 1. Since, it is defined as an illegal value in the above defined SGDC description, the `Soc_10` reports the following violation message for the above design:

```
Node 'top.op1' has illegal value '1' under tag
'use_shi ft', test.sgdc, 2
```

### Example 2

This example illustrates the usage of the `-except` and `-except_type` arguments and the usage of the wildcard characters in the *illegal\_value* constraint:

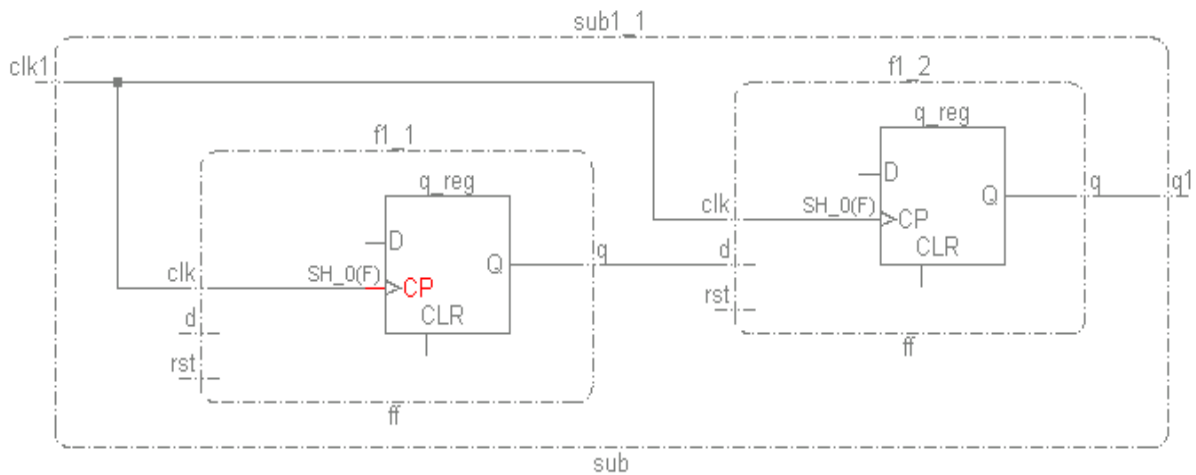
Consider the following sample SGDC file describing the *illegal\_value* constraint description:

```
current_design top
testmode -name top.clk1 -value 0 -scanshift

illegal_value -type FLIP_FLOP_CLOCK -except_type
top.sub1_1.fl_2:TIED_0_SGDC -value 0 -use_shift -
constraint_message_tag m1
```

```
illegal_value -name "top.sub1_1.f*.q_reg.CP" -except
top.sub1_1.f1_1.q_reg.CP -value 0 -use_shift -
constraint_message_tag m3
```

*Figure 2* describes the schematic depicting the violation displayed for the *illegal\_value* constraint having the *-except*, *-except\_type* arguments:



**FIGURE 2.** Violation for the *illegal\_value* constraint

For the above design, for *constraint\_message\_tag* m1, the *Soc\_10* rule reports the following violation message only for *top.sub1\_1.f1\_1.q\_reg.CP* due to the presence of the *except\_type* condition:

```
[constraint_message_tag: m1] Node 'top.sub1_1.f1_1.q_reg.CP
(FLIP_FLOP_CLOCK)' has illegal value '0' under tag '-
use_shift', test.sgdc, 5
```

Also, for *constraint\_message\_tag* m2, the *Soc\_10* rule reports the following violation message only for *top.sub1\_1.f1\_2.q\_reg.CP* due to the presence of the *except* condition

```
[constraint_message_tag: m3] Node 'top.sub1_1.f1_2.q_reg.CP'
has illegal value '0' under tag '-use_shift', test.sgdc, 6
```



### Example 3

Consider the following SGDC description:

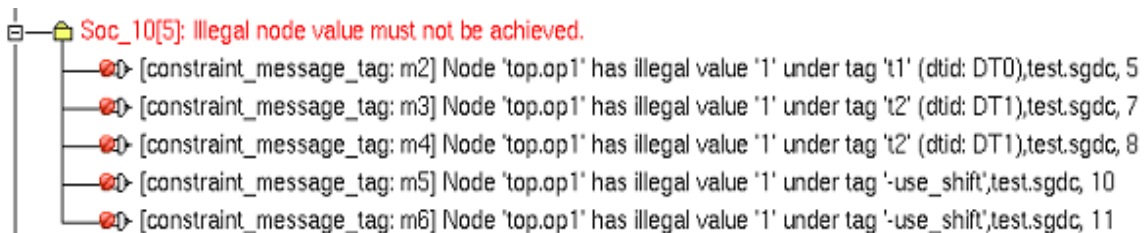
```
current_design top

illegal_value -name top.op1 -value 1 -tag t1
-constraint_message_tag m1
illegal_value -name top.op1 -value 1 -tag t1
-matchNBits 1 -constraint_message_tag m2
illegal_value -name top.op1 -value 1 -tag t2
-constraint_message_tag m3
illegal_value -name top.op1 -value 1 -tag t2 -matchNBits 1
-constraint_message_tag m4
illegal_value -name top.op1 -value 1 -use_shift
-constraint_message_tag m5
illegal_value -name top.op1 -value 1 -use_shift
-matchNBits 1 -constraint_message_tag m6

define_tag -tag t1 -name in1 -value 101
define_tag -tag t1 -name in2 -value 1
define_tag -tag t2 -name in1 -value 1
define_tag -tag t2 -name in2 -value 1

test_mode -name top.in1 -value 101
test_mode -name top.in2 -value 1
```

The Soc\_10 rule reports the following violation message for the message tags, m2, m3, m4, m5, and m6:



**FIGURE 3.** Violation messages for message tag, m5

The Soc\_10 rule does not report violation message for message tag, m1,

because it is defined under tag, t1. This is because the value for t1 is 101, which will be matched against -value field in the *illegal\_value* constraint description for m1.

### **Default Severity Label**

Error

### **Rule Group**

SoC

### **Reports and Related Files**

None

## Soc\_11

**Node must satisfy the specified constraint message tag expression**

### When to use

Use this rule to perform conditional connectivity and value checks on a particular node.

### Description

The Soc\_11 reports violation, if the `constraint_message_tag_expression` specified using the [require\\_constraint\\_message\\_tag](#) is not met on the specified design node.

This rule must be run at the end of all other Soc rules so that message tagging must have happened by then.

### Parameter(s)

None

### Constraint(s)

[require\\_constraint\\_message\\_tag](#) (mandatory): Use this constraint to define the `constraint_message_tag_expression`. The `constraint_message_tag_expression` must be one or a combination of `constraint_message_tags` of the following SGDC commands:

- [require\\_path](#)
- [require\\_value](#)
- [require\\_strict\\_path](#)
- [illegal\\_value](#)
- [illegal\\_path](#)

### Operating Mode

None

## Messages & Suggested Fix:

### Message 1

[ERROR] Node <node-name> does not have required constraint\_message\_tags <constraint\_message\_tag\_expression>. Present: <constraint\_message\_tag>, missing: <constraint\_message\_tag>

#### *Arguments*

- These are nodes name present in the design, <node-name>
- Combination of constraint\_message\_tag\_expression using operator '||' and '&&', <constraint\_message\_tag\_expression>
- Constraint\_message\_tag present and missing, <constraint\_message\_tag>

#### *Potential Issues*

The violation message appears, if at least one constraint\_message\_tag is missing from constraint\_message\_tag\_expression.

#### *Consequence of Not Fixing*

Not fixing this violation may impact the functionality of the design

#### *How To Debug and Fix*

To fix this violation, perform any of the following tasks:

- Check the design for an error
- Check the SGDC file for a missing constraint

### Message 2

[INFO] Node <node-name> has required constraint\_message\_tags <constraint\_message\_tag\_expression>. Present: <constraint\_message\_tag>, missing: <constraint\_message\_tag>

#### *Arguments*

- These are nodes name present in the design, <node-name>
- Combination of constraint\_message\_tag\_expression using operator '||' and '&&', <constraint\_message\_tag\_expression>
- Constraint\_message\_tag present and missing, <constraint\_message\_tag>

#### *Potential Issues*

The violation message appears, if at least one `constraint_message_tag` is missing from `constraint_message_tag_expression`.

### ***Consequence of Not Fixing***

Not fixing this violation may impact the functionality of the design

### ***How To Debug and Fix***

To fix this violation, perform any of the following steps:

- Check the design for an error
- Check the SGDC file for a missing constraint

## **Example Code and/or Schematic**

Consider the following SGDC:

```
current_design top

clock -name clk -testclock
testmode -name rst -value 1

gating_cell -name wb_cgc -clkinTerm clkin -clkoutTerm clkout
-enTerm en -testenTerm te

require_path -from_type INPUT_PORTS -to_type
SCAN_FLIP_FLOP_DATA FLIP_FLOP_DATA LATCH_DATA MUX_SELECT -
constraint_message_tag PORT_CHECK
require_path -from "top.cgc_1.clkout" -to_type
FLIP_FLOP_CLOCK LATCH_ENABLE -constraint_message_tag
CGC_CHECK_1
require_path -from "top.cgc_2.clkout" -to_type
FLIP_FLOP_CLOCK LATCH_ENABLE -constraint_message_tag
CGC_CHECK_2
require_path -from_type LATCH_OUT -to_type FLIP_FLOP_RESET
-constraint_message_tag LATCH_CHECK -report_failure_as_info

illegal_path -from_type BLACK_BOX_OUTPUT -to_type
FLIP_FLOP_DATA -constraint_message_tag BBOX_CHECK -
report_failure_as_info
```

```
require_value -name "top.l_1.out" -value 0
-constraint_message_tag LATCH_VALUE_CHECK
-report_failure_as_info
illegal_value -name "top.d_1.out" -value 1
-constraint_message_tag FLOP_VALUE_CHECK
-report_failure_as_info

require_constraint_message_tag -type LATCH
-constraint_message_tag_expression "LATCH_CHECK:PASS &&
LATCH_VALUE_CHECK:PASS"
require_constraint_message_tag -type LATCH
-constraint_message_tag_expression "LATCH_CHECK:FAIL ||
LATCH_VALUE_CHECK:FAIL"
```

For the above SGDC, the Soc\_11 rule reports the following violation messages as listed in [Table 1](#):

**TABLE 1** Soc\_11 Message Examples

Message	Rule Severity	Description
Node 'top.l_1.temp_reg (LATCH)' does not have required constraint_message_tags 'LATCH_CHECK:PASS    LATCH_VALUE_CHECK:PASS'. Present: 'none', Missing: 'LATCH_CHECK:PASS LATCH_VALUE_CHECK:PASS'	ERROR	In the above violation message the constraint_message_tag_expression, LATCH_CHECK:PASS    LATCH_VALUE_CHECK:PASS, is not met for design node, top.l1. It means neither LATCH_CHECK = PASS nor LATCH_VALUE_CHECK = PASS constraint_message_tags are met. If anyone of them meets then this will be an info message.
Node 'top.l_1.temp_reg (LATCH)' has required constraint_message_tags 'LATCH_CHECK:FAIL    LATCH_VALUE_CHECK:FAIL'. Present: 'LATCH_CHECK:FAIL', Missing: 'LATCH_VALUE_CHECK:FAIL'	INFO	In the above info message the constraint_message_tag_expression LATCH_CHECK:FAIL    LATCH_VALUE_CHECK:FAIL is met for design node top.l1. It means LATCH_CHECK = FAIL constraint_message_tag is met and LATCH_VALUE_CHECK = FAIL constraint_message_tag is not met. As there is logical OR of constraint_message_tags so if anyone of them meets, info message for Soc_11 is reported.

**Default Severity Label**

Info/Error

**Rule Group**

SoC

## Soc\_12

**Node must not have the specified constraint message tag expression**

### When to Use

Use this rule to perform conditional connectivity and value checks on a particular design node.

### Description

The Soc\_12 rule reports violation, if the `constraint_message_tag_expression` specified using the [illegal\\_constraint\\_message\\_tag](#) is met on the design node. This rule must be run at the end of all other Soc rules so that message tagging must have happened by then.

### Parameter(s)

None

### Constraint(s)

[require\\_constraint\\_message\\_tag](#) (mandatory): Use this constraint to define the `constraint_message_tag_expression`. The `constraint_message_tag_expression` must be one or a combination of `constraint_message_tags` of the following SGDC commands:

- [require\\_path](#)
- [require\\_value](#)
- [require\\_strict\\_path](#)
- [illegal\\_value](#)
- [illegal\\_path](#)

### Operating Mode

None



## Messages and Suggested Fix

### Message 1

**[ERROR]** Node <node-name> has illegal constraint\_message\_tags <constraint\_message\_tag\_expression>. Present: <constraint\_message\_tag>, missing: <constraint\_message\_tag>

#### *Arguments*

- These are nodes name present in the design, <node-name>
- Combination of constraint\_message\_tag\_expression using operator '||' and '&&', <constraint\_message\_tag\_expression>
- Constraint\_message\_tag present and missing, <constraint\_message\_tag>

#### *Potential Issues*

The violation message appears if at least one constraint\_message\_tag is missing from constraint\_message\_tag\_expression

#### *Consequence of not fixing*

Not fixing this violation may impact the functionality of the design.

#### *How To Debug And Fix*

To fix this violation, perform any of the following tasks:

- Check the design for an error
- Check the SGDC file for a missing constraint

### Message 2

**[INFO]** Node '<node\_name>' has valid constraint\_message\_tags. Disallowed: '<disallowed\_constraint\_message\_tag\_expression>', Present: '<present\_constraint\_message\_tag>', Missing: '<missing\_constraint\_message\_tag>'

#### *Arguments*

- Name of the design node which does not have the illegal constraint\_message\_tag (check PASSED), <node\_name>
- Disallowed constraint\_message\_tag\_expression, <disallowed\_constraint\_message\_tag\_expression>

- constraint\_message\_tag which are found on the design node, <present\_constraint\_message\_tag>
- constraint\_message\_tag which are missing on the design node, <missing\_constraint\_message\_tag>

### **Potential Issues**

This is an informational message. Therefore, there are no potential issues related to this message.

### **Consequences of Not Fixing**

This is an informational message. Therefore, there are no consequences of not fixing this message.

### **How to Debug and Fix**

This is an informational message. Therefore, no debug or fix is required.

## **Example Code and/or Schematic**

Consider the following SGDC:

```
current_design top
```

```
clock -name clk -testclock
testmode -name rst -value 1
```

```
gating_cell -name wb_cgc -clkInTerm clkIn -clkOutTerm clkOut
-enTerm en -testenTerm te
```

```
require_path -from_type INPUT_PORTS -to_type
SCAN_FLIP_FLOP_DATA FLIP_FLOP_DATA LATCH_DATA MUX_SELECT -
constraint_message_tag PORT_CHECK
```

```
require_path -from "top.cgc_1.clkOut" -to_type
FLIP_FLOP_CLOCK LATCH_ENABLE -constraint_message_tag
CGC_CHECK_1
```

```
require_path -from "top.cgc_2.clkOut" -to_type
FLIP_FLOP_CLOCK LATCH_ENABLE -constraint_message_tag
CGC_CHECK_2
```

```
require_path-from_type LATCH_OUT-to_type FLIP_FLOP_RESET
-constraint_message_tag LATCH_CHECK -report_failure_as_info
```

```
illegal_path -from_type BLACK_BOX_OUTPUT -to_type
FLIP_FLOP_DATA -constraint_message_tag BBOX_CHECK -
report_failure_as_info
```

```
require_value -name "top.l_1.out" -value 0
-constraint_message_tag LATCH_VALUE_CHECK
-report_failure_as_info
```

```
illegal_value -name "top.d_1.out" -value 1
-constraint_message_tag FLOP_VALUE_CHECK
-report_failure_as_info
```

```
illegal_constraint_message_tag -type ICG -
constraint_message_tag_expression "CGC_CHECK_1:PASS ||
CGC_CHECK_2:FAIL
```

```
illegal_constraint_message_tag -type FLIP_FLOP -
constraint_message_tag_expression "BBOX_CHECK:FAIL &&
PORT_CHECK:PASS"
```

For the above SGDC, the *Soc\_12* rule reports the following error message:

```
Node 'top.cgc_2.clkout (ICG)' has illegal
constraint_message_tags 'CGC_CHECK_1: PASS || CGC_CHECK_2: FAIL'.
Present: 'CGC_CHECK_2: FAIL', Missing: 'CGC_CHECK_1: PASS'
```

In the above violation message the `constraint_message_tag CGC_CHECK_1:PASS || CGC_CHECK_2:FAIL` is met for design node `top.cgc_2`. It means in logical or operation of `CGC_CHECK_1:PASS` and `CGC_CHECK_2:FAIL`, the `constraint_message_tag CGC_CHECK_2 = FAIL` is met.

## Default Severity Label

Info/Error

## Rule Group

SoC

## Soc\_14

**Ensure that specified nets are having stable values under specified condition**

### When to Use

Use this rule to detect the sources of instability.

### Description

The *Soc\_14* rule reports violation for nets having unstable values under specified condition.

It reports violation for nodes that have unstable values, specified using the [require\\_stable\\_value](#) constraint.

### Parameter(s)

*dft\_soc\_unstable\_value\_sources*: Default value is none. Set the value of the parameter to `all`, `blackbox`, `hanging_net`, or `port` to specify unstable value sources, other than scannable flip-flops and latches, that needs to be reported by the *Soc\_14* rule.

### Constraint(s)

- [require\\_stable\\_value](#) (mandatory): Use this constraint to specify nodes whose value is expected to be stable.
- [force\\_stable\\_value](#) (optional): Use this constraint to specify nodes that will have a stable value during the scan shift and capture modes.
- [force\\_unstable\\_value](#) (optional): Use this constraint to specify nodes that will have an unstable value during the scan shift and capture modes.

### Operating Mode

None

### Messages and Suggested Fix

[ERROR] <constraint\_message\_tag> Node ' <node\_name>' may not have stable value under <tag> due to possible unstable source(s) <distribution>

#### *Arguments*

- name of the node in the design, <node\_name>
- Condition, <tag>
- Details of unstable sources, <distribution>

### ***Potential Issues***

A violation message is reported when a node has unstable value under a specified condition.

### ***Consequence of not fixing***

Scan flip-flops are considered sources of corruption as their values toggle during the shift and capture mode. The impact of such corruption source propagate through multiple stages of non-scan flops in the fan-in logic cone of the target node (target for stability) until it reaches the node or it is blocked (for example, by the test\_mode constraint).

### ***How To Debug And Fix***

Review the path from unstable sources to the design nodes specified using the [require\\_stable\\_value](#) constraint.

## **Example Code and/or Schematic**

Currently Unavailable

## **Default Severity Label**

Error

## **Rule Group**

SoC

## Atspeed\_21

**Check required pulse pattern at specified node.**

### Rule Description

The Atspeed\_21 rule reports a violation, if the simulated result pattern at a specified node does not match the expected pulse pattern.

### Constraints

*require\_pulse* (Mandatory)

*define\_tag* (Mandatory)

### Rule Parameters

*dftUseOffStateOfClockInClockPropagation*

### Operating Mode

Define\_tag

### Message Details

#### Message 1

[constraint\_message\_tag: <value>] Node <node\_name> (specified as -type) has mismatched value <actual\_pulse\_pattern> between <after\_bit> and <before\_bit> bits under tag <tag\_name>. [REASON: <reason>]. Expected pulse pattern is <expected\_pulse\_pattern>

#### Message 2

[constraint\_message\_tag: <value>] Node <node\_name> (specified as -name) has mismatched value <actual\_pulse\_pattern> between <after\_bit> and <before\_bit> bits under tag <tag\_name>. [REASON: <reason>]. Expected pulse pattern is <expected\_pulse\_pattern>

#### Message 3

[constraint\_message\_tag: <value>] Ignoring require\_pulse constraint as neither -name nor -type field is specified.

## Arguments

- Constraint tag value, <value>
- Name of the net or heir term where pulse is expected, <node\_name>
- Pulse pattern obtained at specified node, <actual\_pulse\_pattern>
- After bit as specified in *require\_pulse* constraint, <after\_bit>
- Before bit as specified in *require\_pulse* constraint, <before\_bit>
- Tag\_name of the define\_tag constraint under which simulation is to be done, <tag\_name>
- Short description of reason for mismatch, <reason>. The following lists the possible reasons for mismatch;
  - No pulse found, in case there are don't care bits present during the pulse
  - Pulse's low-width is too long
  - Pulse's low-width is too short
  - Pulse's high-width is too long
  - Pulse's high-width is too short
  - Mismatch in rear padding
- Expected pulse pattern as specified in the *require\_pulse* constraint, <expected\_pulse\_pattern>

## Location

The file and the line where the *require\_pulse* constraint is specified.

## Schematic highlight

Specified node with the obtained and the expected value.

## Rule Severity

Warning

## Example

The following example illustrates the padding. Consider the following definition of *require\_pulse* constraint:

```
require_pulse -name xyz -tag sim_1\\two pulses required
```



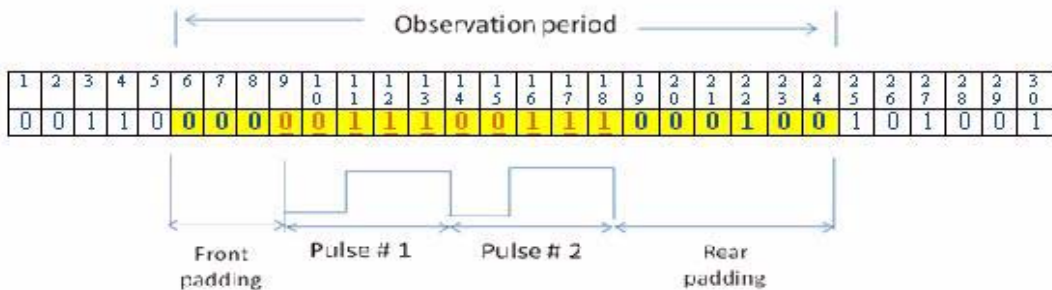
## Overview

```

-pulse 2
-after 5 -before 25 \\observation period
-high_width 3 -low_width 2 \\pulse characteristics

```

Also, consider the following obtained pulse:



The bits in bold are in pulse observation period (-after 5, -before 25). The underline bits are the actual pulses, which are same as described by the [require\\_pulse](#) constraint (2 pulses of 3 high bit and 2 low bit).

However, the above pulse causes a violation because there is a mismatch in rear padding at bit 22. Rear padding starts after complete pulse is matched and ends at the end of observation period. So, there should not be any pulse in rear padding. If bit 22 is 0, then Atspeed\_21 does not report any violation. Instead, the Info\_Atspeed\_21 violation message is generated stating that two pulses are obtained during this period.

In case the `require_pulse` is constraint defined with `-count 2`, `-high_width 3`, and `-low_width 2`, there are following two possible pulses to match depending on whether you start from low bits or high bits:

```

0 0 1 1 1 0 0 1 1 1
1 1 1 0 0 1 1 1 0 0

```

In the above example, during pulse observation window, the first transition is encountered at bit 11. So, bits from 6 to 10 are part of front padding.

Since front padding is sufficiently large to accommodate `low_width`, we assume pulse starts from bit 9 with low bits first and ends at bit 18.

If there are insufficient number of bits in front padding, then bit 11 is considered as pulse starting point.

**NOTE:** *There is no limit on length of front padding or rear padding. However, ensure that there is no transition in the rear padding.*

Also, if the design generates more pattern than expected, then there is a mismatch in rear padding. To avoid the `Atspeed_21` violation, reduce the length of rear padding by adjusting `-before <value>`.

## Info\_Atsped\_21

**Expected pulse pattern at the specified node achieved.**

### Rule Description

The Info\_Atsped\_21 rule reports nodes, which are specified in the following format:

```
require_pulse
  -tag <tagName>
  -name <nodeNames>
  -count <no_of_pulses>
  -after <observe_after_bit_number>
  -before <observe_before_bit_number>
  -high_width <number_of_bits_during_high_phase>
  -low_width <number_of_bits_during_low_phase>
```

Also, the nodes should have the same pulse pattern obtained between after and before bits, when the conditions specified by *<tagName>* are simulated.

### Constraints

*require\_pulse* (Mandatory)

*define\_tag* (Mandatory)

### Rule Parameters

*dftUseOffStateOfClockInClockPropagation*

### Operating Mode

Define\_tag

### Message Details

#### Message 1

[constraint\_message\_tag: <value>] Node <node\_name> (specified as -type) received <number\_of\_pulses> (<pulse\_pattern>) under tag <tag\_name>

### Message 2

[constraint\_message\_tag: <value>] Node <node\_name> (specified as -name) received <number\_of\_pulses> (<pulse\_pattern>) under tag <tag\_name>

### Message 3

[constraint\_message\_tag: <value>] Ignoring require\_pulse constraint as neither -name nor -type field is specified

### Arguments

- Constraint tag value, <value>
- Name of net/heir term where pulse is expected, <node\_name>
- Number of pulses obtained, <number\_of\_pulses>
- Pulse pattern obtained between after and before bits, <pulse\_pattern>
- Tag name of the define\_tag constraint under which simulation is to be done, <tag\_name>

### Location

The file and the line where the *require\_pulse* constraint is specified

### Schematic highlight

The schematic for the Info\_Atsspeed\_21 rule highlights the following:

- Specified node with obtained pulse pattern.
- Expected pulse pattern between after and before bits, specified in the *require\_pulse* constraint on the net
- Actual simulation value between the after and before bits resulting from the simulation of the defineTag condition specified in the *require\_pulse* constraint.

## Rule Severity

Info

## Diagnose\_testmode

**Display instances that block the testmode propagation.**

### When to Use

Use this rule to identify the devices that block testmode signal propagation in shift and capture modes.

### Description

The Diagnose\_testmode rule generates the SpyGlass Explorer highlight information for the devices that block testmode signal propagation.

The Diagnose\_testmode rule generates separate displays for scan shift mode and capture mode.

Testmode simulation is optional. If one or more *test\_mode* constraints are present, then the combinational instances and their blocking pins that block the testmode signal in shift or capture mode are highlighted.

### Default Weight

10

### Language

Verilog, VHDL

### Parameter(s)

- *showPowerGroundValue*: The default value of the parameter is on. Set the value of the parameter to off to hide the simulation value of net due to power/ground.
- *dftShowWaveForm*: The default value of the parameter is off. Set the value of the parameter to on to display the waveform information for the *Diagnose\_testmode* rule. The waveform information is displayed in the SpyGlass Explorer's Waveform Viewer.
- *dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

### Constraint(s)

*test\_mode* (optional): Use this constraint to specify the set of conditions,

both pins and values, that when simulated, will force the circuit in test mode.

## Operating Mode

Scanshift, Capture

## Messages and Suggested Fix

### Message 1

When a blocked testmode signal `<sig-name>` has been found in a design unit `<du-name>` specified as `current_design` under `<mode-name>` mode and the corresponding highlight information has been generated, the `Diagnose_testmode` rule generates the following message:

```
[INFO] Instances through which testmode '<sig-name>' signal doesn't propagate in <mode-name> mode for design '<du-name>' is displayed
```

Where `<mode-name>` can be `shift` or `capture`.

### **Potential Issues**

A violation is reported when other path is not properly sensitized with `test_mode`.

### **Consequences of Not Fixing**

Not fixing the violation may result in improper `test_mode` propagation leading to unscannable flip-flops and reduced coverage.

### **How to Debug and Fix**

For information on debugging, click [How to Debug and Fix](#).

### Message 2

In case, there are no testmode constraints present for design unit `<du-name>` specified as `current_design`, the `Diagnose_testmode` rule generates the following messages depending on the mode for which the testmode constraints are missing:

```
[INFO] Constraint 'testmode -capture' missing in design '<du-name>'
```

```
Constraint 'testmode -shift' missing in design '<du-name>'
```

### **Potential Issues**

A violation is reported when a test mode constraint is missing.

### ***Consequences of Not Fixing***

Not fixing the violation may result in false violation on other SpyGlass DFT rules.

### ***How to Debug and Fix***

View the Incremental Schematic for the violation message. The Incremental Schematic shows the terminal blocking the testmode propagation under the shift/capture mode. Overlay (Auxiliary violation mode) the Info\_testmode rule under the shift/capture mode. This helps in identifying the blocked simulation value.

When the number of nodes reported is more than 100, the Hierarchy button on the Incremental Schematic Window menu bar is enabled. Clicking this button displays the hierarchical representation of the number of nodes reported. For more information on this option, refer to *Viewing Flat Data Hierarchically* section in the *SpyGlass DFT Rules Reference Guide*.

No fix is required as this is an informational rule.

## **Example Code and/or Schematic**

Currently Unavailable

## **Default Severity Label**

Info

## **Rule Group**

Informational Rules

## **Reports and Related Files**

No related reports or files.

## Info\_testmode

### Display testmode simulation results

#### When to Use

Use this rule to display testmode simulation results for both scanshift and capture.

#### Description

The Info\_testmode rule displays all signals with non-x values in testmode scanshift, capture, and functional mode.

**NOTE:** *The functional mode will be simulated on top of 'Power - Ground' mode and it will be displayed when the SpyGlass DFT DSM product is run along with the SpyGlass DFT product.*

The Info\_testmode rule generates one violation message each for scanshift and capture conditions.

The Info\_testmode rule also allows back-referencing from the Schematic Windows to the SpyGlass Design Constraints file (displayed in the Source Window).

#### Prerequisites

The Info\_testmode rule runs only if any power or ground connections exist or if testmode signals are defined.

#### Default Weight

10

#### Language

Verilog, VHDL

#### Method

Simulate power and ground and any available testmode conditions. For each displayed item, the value for that net is back-annotated at the source and at all sinks as follows:

if simulation result = 1 then display "1" on the net

if simulation result = 0 then display "0" on the net

Vectors (multiple bits in the same bus) are merged into a single line on the schematic.



## Parameter(s)

- *showPowerGroundValue*: The default value is on. Set the value of the parameter to off to hide the power/ground simulation values of a net.
- *dftShowForcedValues*: The default value is on. Set the value of the parameter to off to hide the user enforced values, which are displayed with a  $\mathbb{F}$  in the bracket.
- *dftShowWaveForm*: The default value is off. Set the value of the parameter to on to enable the generation of waveform information (displayed in the SpyGlass Explorer's Waveform Viewer) by the Info\_testmode rule.
- *dftUseOffStateOfClockInClockPropagation*: The default value of the parameter is on. Set the value of the parameter to off so that clock lines are kept at X during shift, capture, or atspeed mode simulation.

## Constraint(s)

*test\_mode* (optional): Use this constraint to specify the set of conditions, both pins and values, that when simulated, will force the circuit in test mode.

## Operating Mode

Scanshift, Capture

## Messages and Suggested Fix

### Message 1

[INFO] 'shift mode' simulation value for design '<du-name>' is displayed

### Arguments

To view the list of message arguments, click [Arguments](#).

### Potential Issues

Since this is an informational message, there are no potential issues related to this violation.

### Consequences of Not Fixing

Since this is an informational message, there is no implicit impact of this message.

### ***How to Debug and Fix***

For more information on debugging and fixing the violation, click [How to Debug and Fix](#).

### **Message 2**

[INFO] 'capture mode' simulation value for design '<du-name>' is displayed

### ***Arguments***

To view the list of message arguments, click [Arguments](#).

### ***Potential Issues***

Since this is an informational message, there are no potential issues related to this violation.

### ***Consequences of Not Fixing***

Since this is an informational message, there is no implicit impact of this message.

### ***How to Debug and Fix***

For more information on debugging and fixing the violation, click [How to Debug and Fix](#)

### **Message 3**

In case, no testmode specifications is found for a design unit <du-name>, the Info\_testmode rule generates one of the following messages depending on the mode for which the testmode specifications are missing:

[INFO] Constraint 'test\_mode -capture' missing in design <du-name>

[INFO] Constraint 'test\_mode -shift' missing in design <du-name>

### ***Arguments***

Parent design unit name, <du-name>

### ***Potential Issues***

Since this is an informational message, there are no potential issues related to this violation.

### ***Consequences of Not Fixing***

Since this is an informational message, there is no implicit impact of this message.

### ***How to Debug and Fix***

The `Info_testmode` rule helps in debugging the violations of the `Async_07`, `Clock_11`, and `Latch_08` rules. It is an informative rule, and hence, requires no debug.

Use the `scanshift` rule message when diagnosing scannability problems (for example, the `Clock_11` rule) and the capture rule message when diagnosing capture problems.

Simulation value for the circuit when shift and capture simulation modes are simulated.

When the number of nodes reported is more than 100, the Hierarchy button on the Incremental Schematic Window menu bar is enabled. Clicking this button displays the hierarchical representation of the number of nodes reported. For more information on this option, refer to *Viewing Flat Data Hierarchically* section in the *SpyGlass DFT Rules Reference Guide*.

### **Coloring Scheme**

The `Info_testmode` rule uses the following color scheme in the schematics:

<b>Color</b>	<b>Meaning of the Color</b>
Dark Blue	Net having simulation value 1
Light Red	Net having simulation value 0

No fix is required as this is an informational rule.

### **Example Code and/or Schematic**

Currently Unavailable

### **Default Severity Label**

Info

### **Rule Group**

Information Rules

## Reports and Related Files

No related reports or files.

---

# Appendix: SGDC Constraints

---

## SGDC Concepts

SpyGlass Design Constraints (SGDC) provides additional design information that is not apparent in an RTL.

In addition, you can restrict SpyGlass analysis to certain objects in a design by specifying these objects by using SGDC commands.

## SpyGlass Design Constraints

The following table lists the SGDC commands used by SpyGlass Connectivity Verify product:

<b>SpyGlass Connectivity Verify Product</b>		
<i>atspeed_clock_frequency</i>	<i>clock</i>	<i>define_tag</i>
<i>force_ta</i>	<i>illegal_path</i>	<i>illegal_value</i>
<i>module_pin</i>	<i>require_constraint_message_tag</i>	<i>require_path</i>
<i>require_pulse</i>	<i>require_strict_path</i>	<i>require_structure</i>
<i>require_value</i>	<i>test_mode</i>	

# List of Topics

---

About This Book .....	7
Conditional Connectivity Checks .....	15
Connectivity Checks.....	14
Contents of This Book .....	8
dftAllowNonXValueAtStartOfSensitizedPathInSoc_02 .....	20
dft_allow_path_from_enable_to_cgc_clkout .....	20
dft_conn_check_allow_non_x_value_on_sensitizable_path.....	20
dft_conn_check_allow_trace .....	21
dft_conn_check_handle_rtl_negedge .....	21
dft_connectivity_check_summary .....	31
dft_infer_clock_gating_cell .....	22
dft_max_files_in_a_directory .....	23
dft_require_path_fail_limit.....	23
dft_require_path_invalid_limit .....	24
dft_require_path_pass_limit .....	24
dft_require_path_stop_check_on_pass_count .....	24
dftShowForcedValues .....	25
dftShowWaveForm .....	26
dft_soc_unstable_value_sources .....	25
dft_treat_latches_with_X_on_enable_as_combinational_for_soc_path_checks .....	27
dftUseOffStateOfClockInClockPropagation .....	27
Features of SpyGlass Connectivity Verify product .....	12
Goals in the SpyGlass Connectivity Verify Product .....	18
Licensing Requirements.....	17
Overview.....	12
Overview.....	35
Reports in SpyGlass Connectivity Verify Product .....	30
Require Value Checks .....	13
SGDC Concepts.....	125
showPowerGroundValue .....	28
SpyGlass Connectivity Verify Rule Parameters .....	19
SpyGlass Design Constraints.....	126
Types of Connectivity Verification Checks.....	13
Typographical Conventions .....	9

